Maynooth University
National University of Ireland Maynooth

Feature selection and hierarchical modelling in tree-based machine learning models

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Bruna Davies Wundervald

Under the supervision of:

Prof. Andrew C. Parnell

Dr. Katarina Domijan

Hamilton Institute
National University of Ireland Maynooth
Ollscoil na hÉireann, Má Nuad

March 2023

To my parents and sister, who have given me everything I needed to be the person I am today,

To my Ph.D. supervisors, Andrew Parnell and Katarina Domijan, for the unlimited patience and support, who have made my Ph.D. experience incredibly smooth and enjoyable,

To Walmes Zeviani, Wagner Bonat, and Paulo Justiniano, my undergrad teachers, supervisors, and friends, who gave me the tools and courage to do a Ph.D.,

To Julio Trecenti, Daniel Falbel, Fernando Correa, Athos Damiani, William Amorim and Caio Lente, who are as good statisticians and programmers as they are friends, and who have made an unmeasurable difference in my academic career,

To João Pedro Rimenzoski, Bruno Ritter and Raissa Mariana, who are the best people I ever met, and whom I will love no matter what,

To all my other friends, who have brought joy and love to my life an infinite number of times. I most certainly wouldn't be where I am today if it wasn't for them,

"E tudo que passou, valeu
Pra ver que é real
Paraíso total
É tudo quente e colorido"

Declaration

I hereby declare that I have produced this manuscript without the prohibited assistance of any third parties and without making use of aids other than those specified.

The thesis work was conducted from September 2018 to March 2023 under the supervision of Professor Andrew C. Parnell and Dr. Katarina Domijan in the Hamilton Institute, National University of Ireland Maynooth.

Bruna Davies Wundervald,

Maynooth, Ireland,

March 2023.

Sponsor

This work was supported by a Science Foundation Ireland Career Development Award grant number 17/CDA/4695.

Collaborations

Andrew C. Parnell: As my supervisor, Professor Parnell (Maynooth University) supervised and collaborated on the work of all chapters.

Katarina Domijan: As my supervisor, Dr. Domijan (Maynooth University) supervised the work of all chapters.

Publications

The chapters presented in this thesis have been either published or submitted to peer-reviewed journals. Chapter 3 has already been published in the journal *IEEE Access* and Chapter 4 is currently under review of the *Stat* journal (the reviews have already arrived). Chapter 5 is soon to be submitted to a statistics and machine learning journal. In all papers, Bruna Wundervald is the main author of the paper.

Peer-reviewed journal article:

- B. Wundervald, A. C. Parnell and K. Domijan, (2020) "Generalizing Gain Penalization for Feature Selection in Tree-Based Models", in *IEEE Access*, pp. 190231-190239, 2020, doi: 10.1109/ACCESS.2020.3032095.

Submitted articles (under review):

- B. Wundervald, A. C. Parnell and K. Domijan, (2022). Hierarchical Embedded Bayesian Additive Regression Trees. Under review in the journal *Stats*. *arXiv* preprint: https://arxiv.org/abs/2204.07207.

To Be Submitted:

- Prado, B. Wundervald, A. C. Parnell and K. Domijan, (2023). Hierarchical Embedded Bayesian Additive Regression Trees for Crossed and Nested Random Effects.

Contents

Abstract

Tree-based algorithms are quite popular in the machine learning area in general, due to its many advantages: interpretability, flexibility, high prediction power, and so on. They can be used to many different classification and regression problems, and are in constant development. Because of that, there are many tree-based machine learning algorithms available, including both standard and Bayesian options.

In this thesis, we propose a few methodological extensions to tree-based models including BART, which is the main Bayesian version of it. The list of methods is: extending and generalizing the feature gain penalization idea for tree- based algorithms; extending the BART model into HEBART, to deal with hierarchical data, when there is a grouping variable present; lastly, extending HEBART to deal with more complicated hierarchical data situations. The methods proposed here aim to tackle important deficiencies of the algorithms in question, as they are very popular and in high-demand at the moment.

The first method develops a new gain penalization idea that exhibits a general local-global regularization for tree-based models, which is able to create much more powerful and interpretable generalizations of the gain penalization method. One of the main advantages of this technique is that it can be applied to all (non-Bayesian) tree-based algorithms without loss of generality. The second method switches topics a bit and deals with simple yet powerful extension of Bayesian Additive Regression Trees which we name Hierarchical Embedded BART (HEBART). This model allows for random effects to be included at the terminal node level of the set of regression trees estimated in BART, making it a non-parametric alternative to mixed effects models. At last, we propose yet a few more extensions to HEBART,

namely, I) the Crossed Random Effects HEBART (CHEBART) which allows for multiple grouping variables in the same model; II) the Nested Random Effects HEBART (NHEBART) approach, which accounts for multiple nested grouping variables, where each group level has sub-levels (or sub-groups).

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Non-linear models have been extensively used for regression and classification problems. Trees are a particular case of such models, that recursively partition the feature space, resulting in a local model for each estimated region (Breiman et al., 1984), where the predicted value is a constant, which is numeric for regression settings and a class for classification settings. Such models learn the partitions directly from the training data, creating an adaptive basis function model (ABM) (Murphy, 2012), which can be used to define all tree-based algorithms. Overall, these methods are highly flexible, scale well as the number of samples and features grow and have a competitive prediction power, which makes them an important part of the toolkit of researchers and applied data scientists. Their algorithm simplicity combined with a fairly easy interpretation and wide availability through many packages for different software has made the tree-based models particularly popular, also creating huge research interest in the topic.

There are many variations of tree-based models. We can say that the most popular ones are the CART (Breiman et al., 1984) and Random Forests (Breiman, 2001a), where one is derived from the other. The CART algorithm builds single decision

trees based on a pre-defined cost function to be minimized, in an attempt to uncover the underlying data generation process. Random Forests extend this method by building many trees, where each tree is estimated with a slightly different resample of the original dataset, and not all features are allowed to be examined in each growing step. This technique works because single trees are known to be high-variance estimators, where small changes in the training data can lead to completely different results (Murphy, 2012). Random Forests increase the stability of the predictions, as it uses the property that an average of many estimates has a smaller variance than one estimate, and grow many trees from re-samples of the data. Using bagged ensembles (Breiman, 1996a) such as Random Forests has been proven to be extremely powerful and reliable as a machine learning prediction tool, apart from making minimal assumptions about the data.

As tree-based models became very popular, its Bayesian approaches have also been developed, where the first proposals (Chipman et al., 1998; Denison et al., 1998) focus on algorithms analogous to CART. The Bayesian adaption comes from adding a stochastic element to the fitting of the trees, meaning that all elements of the model are assumed to have prior and posterior probability distributions. In Bayesian analysis, it is already very common to attribute distributions to the response variable, algorithm parameters, shrinkage factors, and so on. However, one of the main novelties of Bayesian trees is the introduction of prior distributions for the tree structures, which are quite complicated to define mathematically, and also heavily data-dependent, but it is an element that makes all the difference in the algorithm fitting.

Bagged approaches have also been created, with the most significant one being the Bayesian Additive Regression Trees (BART), an algorithm based on the sum of trees instead of averages. BART can be thought of as an additive model of the form

$$f(\boldsymbol{x}) = f_1(x_1) + \cdots + f(x_p), \tag{1.1}$$

where each f_j is a tree based on the set of features available and contributes to the creation of the predictions for the response. This implies that the model is fitted

via a back-fitting algorithm, as each tree is estimated using the set of residuals from the prediction created by the sum of all the other trees. The full algorithm is called iterative Bayesian back-fitting Markov Chain Monte Carlo algorithm (Hastie and Tibshirani, 2000; Brooks et al., 2011a), where all the hyperparameters are sampled from their posterior distributions, and MCMC is applied where necessary. There are priors for many aspects of the trees, such as the the tree structure, tree depth, and terminal node parameters, which is one of the main reasons BART usually is highly adaptive and has a good predictive performance.

Since it has been proposed, BART has drawn attention from many areas, especially after so many computational resources have become widely available. There have been plenty of applications of BART, including credit risk modelling (Zhang and Härdle, 2010), causal inference (Hill, 2011; Green and Kern, 2012; Hill and Su, 2013), survival analysis (Bonato et al., 2011), spam-detection (Abu-Nimeh et al., 2008), and time-series analysis (Prüser, 2019; Clark et al., 2021). BART has also been used in many health-related areas, such as proteomic discovery (Hernández et al., 2018), hospitals' evaluation (Liu et al., 2015), preeclampsia and stillbirth risk (Starling et al., 2019, 2020), and treatment effects for causal analyses (Santos and Lopes, 2018). The many extensions to BART include a version to account for polychotomous response (Kindo et al., 2016b), multivariate skewed responses (Um, 2021), density regression (Orlandi et al., 2021), count/semi-continuous zero-inflated data (Linero et al., 2020a; Murray, 2021), high-dimensional data (Linero and Yang, 2018a; He et al., 2019), and heteroscedastic data (Pratola et al., 2020b), as well as quantile regression (Kindo et al., 2016a), and semi-parametric models (Zeldow et al., 2019; Tan and Roy, 2019). Varying coefficient models have also been developed (Deshpande et al., 2020), which tend to create a more sophisticated BART model.

This thesis presents extensions to both tree-based methods in general and specifically for BART: i) the first proposed method is for overcoming a limitation of tree algorithms, namely its inability to automatically perform feature selection, and ii) a few different extensions to model hierarchical data with BART, e.g. when we have one or more grouping variables and are interested in accounting for their effect on the process that generates the data.

The first method proposed a generalization of gain penalization in tree-based models for feature selection. This generalization is made in two senses: for the penalization methodology and in the algorithm type. For the algorithm, this means that the regularization method can be applied to any tree-based algorithm, be it either single trees such as CART, or elaborated ensembles like Bagging and Random Forests. As for the penalization method, we generalize it by proposing a gain penalization based on a weighting parameter combined with a certain function, usually based on some characteristic of the feature or its relationship to the response. As it will be detailed later on, we propose a local-global form of penalization, where the equation balances how much all features should be jointly penalized and how much will it be due to a local function that is manually set. This formulation has inspiration from the use of priors made in Bayesian methods since we intend to introduce prior knowledge regarding the importance of each feature in the model.

In a second scenario, the BART extensions for hierarchical data merges the ideas from traditional Bayesian hierarchical modeling (Gelman and Hill, 2006) and linear mixed effects models (Pinheiro and Bates, 2000) with BART. Overall, we allow the trees to have extra splits on each terminal node, corresponding to the specification of the grouping variables. Thus we introduce intra-group node parameters in BART, and we refer to these parameters as the sub-terminal node levels. The new parameters allow us to have a group-specific prediction for each node, and the usual overall terminal node prediction. The flexibility of this structure means that there is no requirement for the user to specify where the random effect is included, for example as an intercept or as a regression slope. These extensions open up a multitude of possibilities for BART, as they can be applied to longitudinal data, repeated measures data, multilevel data, and block designs, to cite a few.

1.2 Outline of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, we provide some background on the tree-based algorithms in general, make definitions and set the notation used throughout the thesis. We present some terminology and how trees are learned under the CART algorithm, discuss the Random Forests algorithm

and some of its specific limitations, and then move on to explaining the BART theory. The following chapters are composed of three journal articles created for this thesis, the first one being more general on tree-based models and the two other ones more specific to BART. The remainder of the thesis explains our conclusions and possible extensions to the methods proposed here.

In Chapter 3, we extend and generalize the feature gain penalization idea for tree-based algorithms (Deng and Runger, 2013). In the first moment, we show how the previous methods do not perform sufficient regularization and tend not to select the best features possible. Such methods often exhibit sub-optimal out-of-sample performance, especially when correlated features are present, which is one of the main problems we want to tackle. Instead, we develop a new gain penalization idea that exhibits a general local-global regularization for tree-based models, which is able to create much more powerful and interpretable generalizations of the gain penalization method. This new technique allows for full flexibility in the choice of feature-specific importance weights, while also applying a global penalization to all the features. We validate our method on both simulated and real datasets, exploring how the hyperparameters interact with the final fit of the model. For the implementation, we provide our method as an extension of the popular `R` (R Core Team, 2018) package `ranger` (Wright and Ziegler, 2017), a package that is most commonly used for random forests but that is able to run any of the tree-based methods mentioned here.

Chapter 4 changes the topic a bit, and proposes a simple yet powerful extension of Bayesian Additive Regression Trees which we name Hierarchical Embedded BART (HEBART). As briefly discussed before, the proposed model allows for random effects to be included at the terminal node level of the set of regression trees estimated in BART. This simple addition makes HEBART a non-parametric alternative to mixed effects models which avoids the need for the user to specify the structure of the random effects in the model. One of the main advantages of our model is that it is able to maintain the prediction and uncertainty calibration properties of standard BART. Using simulated and real-world examples, we demonstrate that this new extension yields superior predictions for some of the standard mixed effects models' example data sets, and yet still provides consistent

estimates of the random effect variances. We also provide the full implementation as an `R` (R Core Team, 2018) package and the code for the experiments shown here.

Finally, Chapter 5 discusses two extensions to our own HEBART algorithm. The first one is the Crossed Random Effects HEBART (CHEBART) which allows for multiple grouping variables in the same model, where a proportional number of trees uses each of such variables. The second one is the Nested Random Effects HEBART (NHEBART) approach, that accounts for multiple nested grouping variables, where for each level of the groups have sub-levels (or sub-groups). These two extensions are analogous to some of the more popular model structures used in LME, which also allows for crossed and nested random effects, as those are quite common modeling tasks. In the mentioned section, we explain the mathematical definitions of each model and provide their corresponding proposed algorithms. Their performance on both simulated and real data sets that are commonly used in the traditional hierarchical modeling literature is demonstrated and it is satisfactory for both cases. In addition, both models can better handle wrong specifications on the model structure, in comparison to standard models such as linear mixed models, and do not easily yield estimation errors. For both algorithms, we also provide the full implementation as `R` (R Core Team, 2018) packages, which we intend to aggregate into the main HEBART package, concentrating all similar models in only one source.

CHAPTER 2

Tree-based algorithms

*I*n this chapter, we will present and discuss the main features of the most common tree-based algorithms. We first introduce decision trees under the CART recursive partitioning algorithm, which was one of the first tree-based models created. Then we move on to the Random Forests algorithm, one of the most common examples of a tree-based bagged model and the main basis for the topic of the following chapter. Finally, we explain the Bayesian version of trees and its correspondent BART algorithm, which composes the theme of the following chapters of this thesis.

2.1 Tree-based algorithms

Tree-based algorithms are a particular case of non-linear models with adaptive basis functions that recursively partition the feature space, resulting in a local model for each estimated region (Breiman et al., 1984). To better explain what this means in practice, let us define the features set as a matrix with p columns and n rows, where each i-th row is composed by the p-dimensional vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, where $i = 1, \ldots, n$. We also define the target variable as a vector of length n, composed by the samples $y_i \in \mathbf{y}$.

Now, the best approach to explaining a tree-based algorithm and its the growing process is graphically. Consider Figure 2.2, which depicts a simple decision tree. In panel (a) of the image, we can see that the root node of the tree captures the rule 'is the value of feature x_1 less than a certain defined threshold t_1?'. If yes, the x_1 samples that satisfy this condition will be allocated into the left child of this node, while the other will be allocated to the right node. This creates two new feature regions, called R_1 and R_2. On the next step R_2 is separated into two regions, which are based on the decision of whether x_2 is less than threshold t_2, generating two nodes that define R_2 and R_3. For a regression case, which is our focus on this thesis, a mean response is associated to each of the estimated regions, resulting in a piecewise constant surface, shown in panel (b) of the same figure. We can see what this procedure does to the feature space, breaking it into smaller regions with their own prediction value. With this, we can more formally define a tree-based algorithm as a function:

$$f(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] = \sum_{r=1}^{R} w_r \mathbb{I}(\mathbf{x} \in R_r) = \sum_{r=1}^{R} w_r \phi_r(\mathbf{x}; \mathbf{v}_r), \tag{2.1}$$

where R_r is the r-th region, w_r is the prediction given to this region, ϕ_r is the r-th basis function that is learned from the data, and $\mathbf{v}_r$ represents the splitting feature chosen and the corresponding splitting value.

These rules can be estimated using a greedy procedure that computes a locally optimal maximum likelihood estimator by finding the splits that lead to the minimization of a cost function. For regression, the cost function of a decision $\mathbb{D}$ is frequently defined as $\text{cost}(\mathbb{D}) = \sum_{i \in \mathbb{D}} (y_i - \bar{y})^2$, where $\bar{y} = (\sum_{i \in \mathbb{D}} y_i) |\mathbb{D}|^{-1}$ is the mean of the observations in the specified region. The tree continues to deepen until some stopping criterion is reached, usually if there are no more significant changes in the cost function given by the addition of new splits.

The generalization to classification settings comes from replacing the mean response by an empirical distribution of class labels in each terminal node. Still considering Figure 2.2, this algorithm could also be used for predicting a class target, where the predicted value in each region would be the most common class. With

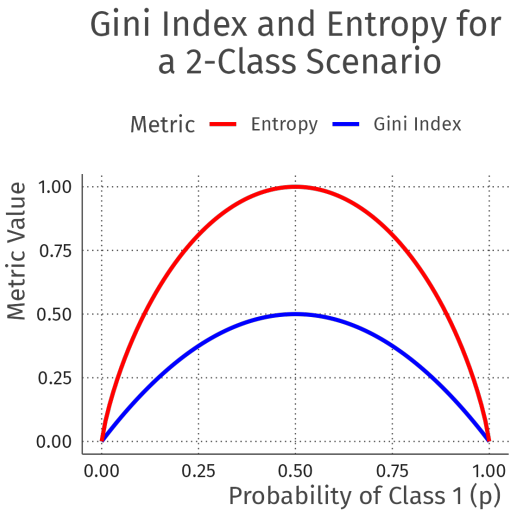


Figure 2.1: Gini Index and Entropy functions for a 2-class scenario: both functions are sensitive to class probabilities changes.

this, the corresponding cost function switches to $\text{cost}(\mathbb{D}) = 1 - \frac{1}{|\mathbb{D}|}\sum_{i\in\mathbb{D}}\mathbb{I}_{(y_i=c)}$, where $\mathbb{D}$ represents the samples in each terminal node and and c is a target class. This function is called the misclassification rate and is what we try to minimize when fitting such an algorithm. As the classification setting has sparked more interest in the machine learning community, there are a few more options for cost functions, including the entropy: $\mathbb{H}(\hat{\pi}) = -\sum_{c=1}^{C}\hat{\pi}_c\log(\hat{\pi}_c)$, which is a very common function in information theory, and even the Gini index, defined as $\text{Gini} = 1 - \sum_c \hat{\pi}_c^2$. In Figure 2.1, we have an example of both functions, where we can see that, in a 2-class scenario, the Gini Index and Entropy are sensitive to class probabilities changes, meaning that they are more reliable is most settings. In contrast, the error rate can easily produce very similar results for very different fits, where one might be preferable over the other, often leading to sub-optimal solutions.

The above approach is known as a CART model (Breiman et al., 1984). Tree-based algorithms are very popular both in the statistics and machine learning communities due to their many advantages. The algorithms are usually not computationally expensive and the results are logically interpretable. The binarization of decisions

in a tree is even thought to be similar to the way humans make decisions, which gives the trees an extra appeal. However, the simplest models also come with a few drawbacks. For instance, CART often has lower predictive power when compared to other popular algorithms and is usually not able to perform feature selection very well. A more dramatic issue is its instability in relation to small changes in the data, leading to completely different fits and interpretations of the model. More powerful approaches to tree-based models have been proposed and are now widely used, such as the Random Forest algorithm, which we will discuss in the following section.

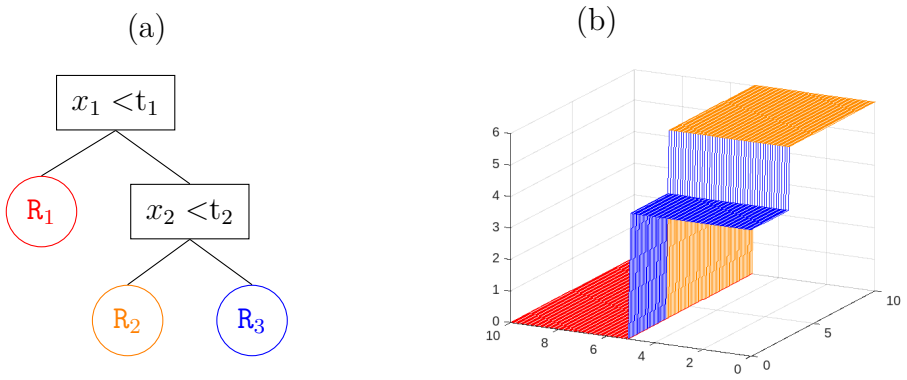


Figure 2.2: An example of a decision tree and its actions in feature space. In panel (a) of the image, we can see that the root node of the tree contains the rule 'is the value of feature x_1 less than a certain defined threshold t_1?'. If yes, the x_1 samples that satisfy this condition will be allocated into the left child of this node, while the other will be allocated to the right node. Panel (b) shows how this procedure splits the feature space into disjoint regions, each with its own predicted value.

2.2 Random Forests

Single CART trees are famous for being high-variance estimators. One approach to overcome the high variance issue is to use a bagged ensemble, which combines many different decision trees in a pre-specified way. More specifically, the Random Forests algorithm uses the mean of the tree estimators as its final prediction, or the majority vote of all trees estimated in a classification setting.

Random Forests also contain other differences in comparison to CART. First, they use a set of bootstrap resamples of the data (chosen randomly and with replacement) instead of the original set, so each fit will be performed in datasets slightly different from the starting one. Second, they only allow a subset m of the features to be the candidates in each tree split, avoiding the creation of excessive correlation between the estimated trees. The final estimator can be described by the bagged ensemble of the form (Breiman, 1996b)

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N_{tree}} \frac{1}{N_{tree}} \hat{f}_n(\mathbf{x}), \tag{2.2}$$

where $\hat{f}_n$ corresponds to the prediction from the n-th tree.

All these alterations are made such that the Random Forest algorithm will produce more reliable and accurate results, which has been often seen to be true in the literature. There are many examples of the successful application of Random Forests in the most diverse fields (e.g. Goldstein et al., 2010, 2011; Pauly, 2012; Ziegler and König, 2014; Alexander et al., 2014; Belgiu and Drăguţ, 2016; Cano et al., 2017), and the original paper (Breiman, 2001a) has received over 100,000 citations in approximately 20 years, even more than the older CART paper (Breiman et al., 1984).

There are fewer papers discussing the properties of Random Forests, such as its calculation of feature importance values. To explain what this metric represents, let us first define the gain of a new split in a single tree as a normalized measure of the cost reduction, given by:

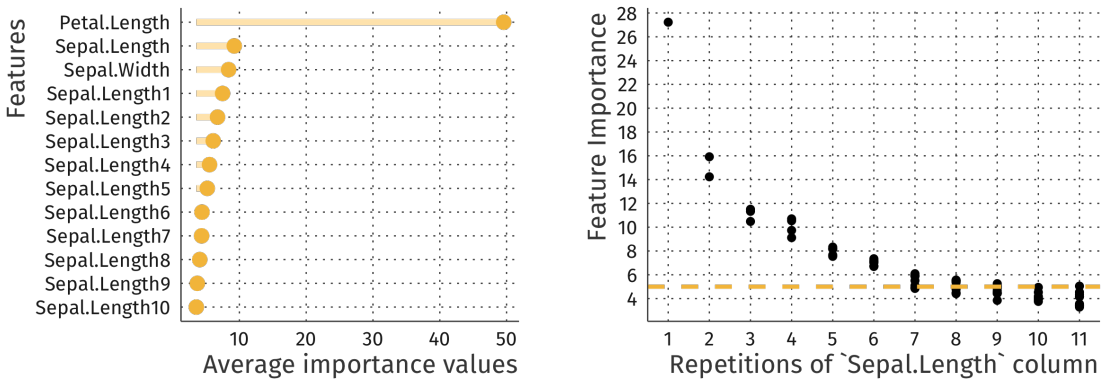
$$\begin{aligned} \Delta(i,t) = \text{cost}(\mathbb{D}) - \\ \left(\frac{|\mathbb{D}_{LN_{(i,t)}}|}{|\mathbb{D}|} \text{cost}(\mathbb{D}_{LN_{(i,t)}}) + \frac{|\mathbb{D}_{RN_{(i,t)}}|}{|\mathbb{D}|} \text{cost}(\mathbb{D}_{RN_{(i,t)}}) \right), \end{aligned} \tag{2.3}$$

for feature i at splitting point t, while $\mathbb{D}$ is related to the previously estimated split, with $LN =$ (left candidate node) and $RN =$ (right candidate node). In tree ensembles, the feature importances values are averaged over all the trees so that:

$$\text{Imp}_i = \frac{1}{N_{tree}} \sum_{n=1}^{N_{tree}} \Delta(i)_n, \tag{2.4}$$

where the global importance value for a feature i is given by accumulating the gain $\Delta(i) = \sum_{t \in \mathbb{S}_i} \Delta(i,t)$, where $\mathbb{S}_i$ represents all the splitting points used in a tree for the ith feature.

The issue here is that performing feature selection and dimensionality reduction in Random Forests becomes non-trivial, as the feature importance metric is not always reliable. A common problem is the underdetection of noisy or correlated features, which can badly influence the results (Strobl et al., 2008). Consider, for example, the results depicted in Figure 2.3, which corresponds to the fit of a random forest algorithm to the popular *Iris* dataset (Anderson, 1936), where the target variable is the classification in one of the flower species (*setosa*, *versicolor* or *virginica*), based on sepal length and width, and petal length and width. For demonstrating the feature selection issue, we also added 10 repetitions of the *Sepal.Length* feature to introduce multicollinearity to the data. In part (a) of Figure 2.3, we can see that feature importance gets divided into the replicas of *Sepal.Length*, when ideally the algorithm should have attributed a high importance to only one of such features, as they are exactly the same. This problem is a natural consequence of the way Random Forests are built, as it has no strategy to detect whether features are too similar and the algorithm will use whatever data it has been fed. In part (b) of the same Figure, where we have as many points in the plot as there are repetitions of the *Sepal.Length* variable in the model, we see how the importances decreases for each variable when this number of repetitions increases, making the feature importance split even more evident. If this model were good at picking up on highly correlated features, there should only be one repetition of *Sepal.Length* being used (or, in other words, having the highest importance value) in the final algorithm, but that is not what happens at all. In such a situation, if we were performing feature selection by hand having a certain importance threshold as a reference, we would probably be selecting way too many correlated features and bringing irrelevant information into our conclusions when doing inference about this model.



(a) (b)

Figure 2.3: Demonstration of feature importance behavior in the presence of multicollinearity. In (a) we have the average feature importance for each feature, when there are many repetitions of *Sepal.Length* in the model. There is a clear split in the importance of the features, when ideally we should have only one of them as the most important one. In (b), we can see how the importances decrease when the number of *Sepal.Length* repetitions increase. In this case, if we were to set a feature importance threshold to discard features, the results could be highly misleading.

Naturally, the feature selection problem has already been studied and discussed by a few authors. For example, in Deng and Runger (2012), the authors first discuss the regularization of Random Forests by gain penalization. The *Regularized Random Forest* (*RRF*) is a way of weighting the gains of each tree split during the growing part of the algorithm. In this setting, the regularized gain is defined as

$$\text{Gain}_R(\mathbf{X}_i, t) = \begin{cases} \lambda\Delta(i,t),\ i \notin \mathbb{U} \text{ and} \\ \Delta(i,t),\ i \in \mathbb{U}, \end{cases} \tag{2.5}$$

where $\mathbb{U}$ is the set of indices of the features previously used, $\mathbf{X}_i$ is the candidate feature, and t the candidate splitting point. The $\lambda \in (0, 1]$ hyperparameter is the penalty coefficient that controls the amount of regularization each feature receives. When this penalization is applied a feature gain is weighted down only if the feature is new to the whole ensemble. The gain penalization technique has a memory of which features were already used in the algorithm, as it tries to avoid using new

features in deeper splits.

In the initial papers λ was treated as a constant value for all the features but this is quite crude and might not perform optimally. In an ideal setting, there should be a regularization parameter for each feature that represents or is connected to the information they carry about the target. In a more recent paper (Deng and Runger, 2013), the authors modify the gain penalization idea by introducing the *Guided RRF*. This method consists of first running a Standard Random Forest (`mtry` $\approx \sqrt{p}$, number of trees = 500) and producing an importance measure for each feature and scaling this measure, in order to find $Imp_i^{'} = \frac{Imp_i}{max_{j=1}^{P} Imp_j}$ where Imp_i is the importance measure calculated for the i-th feature in the Random Forest. Now, the normalized value defined as $Imp_i^{'}$ becomes a component of the penalization factor, in the form $\lambda_i = (1-\gamma)+\gamma Imp_i^{'}$, being now a mixed weighting function.

The ingenuity of this new technique relies on the introduction of a feature-based penalization that now depends on more informed values. However, the usage of a random-forest based $Imp_i^{'}$, or the feature-based part of the penalization function, comes with a few problems. This initial random forest will bring with it the biases we discussed above, such as giving similar importance to highly correlated features. In addition, the paper does explore other methodologies or extensions, such as the influence of the Random Forests hyperparameters including the number of trees and the number of variables to apply to each split.

This is the main motivation for the work we propose in Chapter 3. This chapter explains a generalization of gain penalization in tree-based models for feature selection, where we extensively explore the properties of our generalization.

2.3 Bayesian Additive Regression Trees

The initial Bayesian Regression Tree model was first proposed over 20 years ago (Chipman et al., 1998), and consists of an algorithm that fits a set of CART decision trees using Bayesian inference. Figure 2.4 shows a simple decision tree based on a single rule $x_1 < t_1$. In Bayesian CART, instead of receiving one single predicted value, each terminal node is assumed to have a full probability

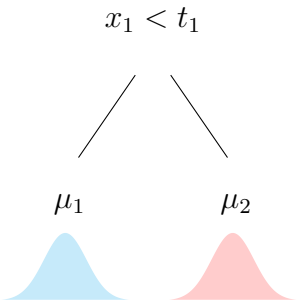


Figure 2.4: A split in a Bayesian tree, where each node has its own probability distribution.

distribution. In the image, the center parameters for the node distributions are μ_1 and μ_2, and the scale parameters can either be varying or be the same for all nodes. While the main idea of this model remains very similar to standard CART, the fitting algorithm follows common Bayesian practices. All parameters, including the tree structures, are sampled from posterior distributions, which can be closed-form (when they are available) or sampled via Markov Chain Monte Carlo methods (Brooks et al., 2011b).

However, in this Chapter we will focus more on the more advanced Bayesian Additive Regression Trees (Chipman et al., 2010b) approach. This model assumes that the generating system of a continuous random variable $\mathbf{y} = [y_1, \ldots, y_n]$ can be approximated by a sum of regression trees, which increases its prediction capacity. In this model, the root nodes do not simply have a predicted value, but instead are assumed to have a probability distribution (the same as BCART), as do all the other elements of the algorithm, but the final posterior for each observation is composed of a sum of distributions instead of a single value. Mathematically speaking, in a standard regression setting the BART model is usually written as:

$$y_i = \sum_{p=1}^{P} \mathbf{G}(X_i; \mathcal{T}_p, \Theta_p) + \epsilon_i, \ \epsilon_i \sim N(0, \tau^{-1}), \tag{2.6}$$

for observations $i = i, \ldots, n$, where P is the fixed number of trees, X_i represents the set of covariates; $\mathcal{T}_p$ is a tree structure, and Θ_p represents a set of terminal node parameters. The function $\mathbf{G}$ maps the tree structures into the set of covariates

X_i and returns a terminal node prediction from Θ, by passing all the values in the vector X_i through the tree structure T_p to get the predicted value. Note that now we have a sum of trees instead of an average, which would be the case in the random forests algorithm. Similar to a bagged ensemble, each tree still behaves as a weak learner, but we are now working with a sum of probability distributions instead of an average of any form. In a regression case, the set of node parameters Θ_p consists of a of $\mu_{p,l}$ parameters for each of the $l = 1, \ldots, L_p$ terminal nodes in tree p. These values provide the tree-level predictions, which are summed together to create an overall predicted value. In general, the residual term ϵ is assumed to follow a Normal distribution with residual precision τ.

We write the set of all trees and parameters as $\mathcal{T}$ and Θ respectively. The joint posterior distribution of the trees and all the parameters is then given by:

$$P(\mathcal{T}, \Theta, \tau | \mathbf{X}, \mathbf{y}) \propto \left[\prod_{p=1}^{P} \prod_{b=1}^{b_p} \prod_{i:\mathbf{x}_i \in \mathcal{D}_{p,b}} p(y_i | \mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau) \right] \times \left[\prod_{p=1}^{P} \prod_{b=1}^{b_p} p(\mu_{p,b} | \mathcal{T}_p) p(\mathcal{T}_p) \right] p(\tau), \tag{2.7}$$

where $p(y_i | \mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau)$ is the normally distributed likelihood as defined in Equation 5.1, and $\mathcal{D}$ represent the regions in the features space (e.g. tree nodes) created by each tree (as demonstrated in Figure 2.2). The term $p(\mu_{p,b} | \mathcal{T}_p)$ is the prior distribution on the terminal node parameters across each terminal node b in each tree p, which is usually taken to be a Normal distribution, leading to a Normal posterior as well. In addition, $p(\mathcal{T}_p)$ is the prior distribution on the tree structure (no closed-form, as the number of possible tree structures grows rapidly), and $p(\tau)$ is the prior distribution on the residual precision, commonly taken as a Gamma distribution.

As we have mentioned before, some of the elements of the algorithm can be quite fiddly to understand. One example is the prior distribution on the trees proposed by Chipman et al. (2010b), which involves applying a separate probability term for each node in the tree. Each of those terms considers both the probability of a split as well as the probability of a new splitting variable being chosen. With this, for an entire tree $\mathcal{T}_p$, it is common to set:

$$P(\mathcal{T}_p) \propto \prod_{\eta \in L_{\mathcal{T}}} (1 - P_{SPLIT}(\eta)) \prod_{\eta \in L_{\mathcal{I}}} P_{SPLIT}(\eta) \prod_{\eta \in L_{\mathcal{I}}} P_{RULE}(\rho|\eta), \tag{2.8}$$

where $L_{\mathcal{T}}$ and $L_{\mathcal{I}}$ represent the sets of terminal and internal nodes, respectively, and ρ represents a generic splitting value. The probability of a node being non-terminal is given by $P_{SPLIT} = \alpha(1 - d_\eta)^{-\beta}$, where d_η denotes the depth of the node η. The recommended values for the hyperparameters are $\alpha \in (0, 1)$ and $\beta > 0$, which control the depth and 'bushiness' of the trees. For the probability of new splits being added to the tree, we have $P_{RULE}(\rho|\eta, \mathcal{T}) = \frac{1}{p_{adj}(\eta)} \frac{1}{n_{j.adj}(\eta)}$, where $p_{adj}(\eta)$ represents how many predictors are still available to split on in node η, and $n_{j.adj}(\eta)$ represents how many values in a given predictor are still available (consequently, one value depends on the other).

The joint prior distribution for the terminal nodes and overall parameters in the standard regression case is denoted by $p(\Theta, \mathcal{T})$. This distribution is created given a prior with a standard conjugate form:

$$\mu_1, \ldots, \mu_b | \tau_\mu, \mu_\mu, \mathcal{T} \sim \mathcal{N}(\mu_\mu, \tau_\mu^{-1}),$$

where τ_μ^{-1} and μ_μ are chosen such that a high density of this distribution is apportioned to the range $[y_{min}, y_{max}]$ interval, by setting $P\mu_\mu - k\sqrt{P(\tau_\mu^{-1})} = y_{min}$ and $P\mu_\mu + k\sqrt{P(\tau_\mu^{-1})} = y_{max}$, for some value of k. This strategy is in place due to the idea that $E[Y|X]$ is very likely within the $[y_{min}, y_{max}]$ interval, so the induced prior reflects that. In addition, the response variable y is usually standardized before the model is run, allowing for reasonable guesses as to the hyper-parameter values of μ_μ and τ_μ, though these too can be estimable parameters.

The residual precision prior is set as $\tau \sim \text{Gamma}(\nu/2, \gamma\nu/2)$, where γ and ν are fixed beforehand. An oft-used tactic is to set the two hyper-parameters such that the BART residual precision has a high probability of being greater than an equivalent precision value from a standard linear regression model applied to the same data (quantiles such as 0.75, 0.90 and 0.99 are frequently used). The idea

behind this prior setup is to put a high density on precision values that would at least be better than a linear model.

Since its creation, BART has been applied in a wide variety of different application areas. The model has been shown to be useful for credit risk modeling (Zhang and Härdle, 2010), survival data analysis (Sparapani et al., 2016, 2020), ecology and evolution modelling (Carlson, 2020), weather and avalanche forecasting (Blattenberger and Fowles, 2014), and genetics (Waldmann, 2016). A popular approach is its use in causal inference (Hill, 2011; Hahn et al., 2020), where BART produces accurate estimates of average treatment effects and is competitive even with the true data generating model.

Beyond applications, many fundamental extensions to the standard BART model have been proposed. Some of the first include adapting BART for categorical, count, and multinomial regression (Murray, 2021; Kindo et al., 2016b) and quantile regression (Kindo et al., 2016a). This was followed by the proposal of BART that adapts to smoothness and sparsity (Linero and Yang, 2018b), models for high-dimensional data and variable selection (Linero, 2018a), BART for zero-inflated and semi-continuous responses (Linero et al., 2020b) and an extension proposed by Hernández et al. (2018), where the authors combine BART with Bayesian Model Averaging to obtain posterior distribution more efficiently when there is a large number of variables available. More recently BART has been extended for use with heterocedastic data (Pratola et al., 2020a), for the estimation of monotone and smooth surfaces, (Starling et al., 2020), varying coefficient models (Deshpande et al., 2020), semiparametric BART (Prado et al., 2021b), and a combination of BART with model-trees (Prado et al., 2021a). As this is a fairly new class of machine learning algorithm, progress on the theoretical performance of BART has only just begun. Some of the mathematical properties of BART, including a deep review of the BART methodology can be found in Linero (2017), and some more general theoretical results in Ročková and van der Pas (2020); Ročková and Saha (2019).

2.3.1 The BART fitting algorithm

The BART model is fully estimated via a backfitting MCMC algorithm (Hastie and Tibshirani, 2000), which holds all other trees constant while the current one is being updated, considering that the final model is based on a sum of trees. This involves calculating the full conditional distribution $P(\mathcal{T}_p, \Theta_p|\tau, \mathbf{X}, \mathbf{y}, \mathcal{T}_{(p)}, \Theta_{(p)})$, where $\mathcal{T}_{(p)}$ represents the set of all trees *except* for tree p (here, the definition of $\Theta_{(p)}$ is analogous). With this strategy, the conditional distribution depends on $(\mathbf{X}, \mathbf{y}, \mathcal{T}_{(p)}, \Theta_{(p)})$ only via the current state of the residuals, defined as

$$R_{i,p} = y_i - \sum_{t \neq p} \mathbf{G}(X_i; \mathcal{T}_l, \Theta_l),$$

meaning these partial residuals include the sum of the predictions for all trees except for tree p. The choice of prior distributions on the trees and terminal nodes allows for the term $P(R_p|\mathcal{T}_p, \tau)$ to be calculated in closed form, naturally avoiding the need for trans-dimensional MCMC methods, which greatly simplifies the resulting fitting algorithm.

BART identifies optimal trees in a completely different way than that of standard decision tree models. Due to the lack of a closed-form posterior distribution, the new trees are proposed using a Metropolis-Hastings sampler (Brooks et al., 2011b). Each candidate tree is obtained via one of the 4 available steps: the GROW move, where a terminal node is selected uniformly from the set of terminal nodes and split into two, with a new split variable and split value chosen analogously (uniformly); the PRUNE move, where a pair of terminal nodes with a common parent are collapsed together and that node stops existing in the tree; the CHANGE move, where a splitting rule is chosen uniformly across the tree and is changed to a new split variable and split value; and finally, a SWAP move, where a parent-child pair of internal nodes is chosen uniformly and swapped in the tree. Naturally, the most common steps are GROW and PRUNE, which are usually more likely to result in an accepted move. However, the movement choice requires probabilities for each move, that can be equal or depend on prior beliefs about moves that should be prioritized in the algorithm (e.g. it might not be efficient to give equal probabilities

to all move types).

In the GROW move, the means by which the splitting variable and value are chosen are slightly different across the implementations of BART in the literature. For instance, instead of sampling everything uniformly, we can put a Dirichlet hyperprior on the splitting features and rules (Linero, 2018b), such that some features are more/less likely than others. With this, the algorithm can keep track of which features are most useful for splits and avoid using too many variables.

In summary, the BART algorithm samples from $P(\mathcal{T}_p, \Theta_p|\tau, R_p)$ via two main steps:

1. Proposes a new tree through one of 4 proposal moves and calculate $P(\mathcal{T}_p|\tau, R_p) \propto P(\mathcal{T}_p)P(R_p|\mathcal{T}_p, \tau)$, and
2. Sample a new set of terminal node parameters via $P(\Theta_p|R_p, \mathcal{T}_p, \tau)$, for the new tree

However, the act of accepting or rejecting a tree move depends on its likelihood in relation to the previous tree. The MCMC algorithm allows us to evaluate and compare those probabilities, and stochastically decide on which trees to accept or reject. The probability $P(\mathcal{T}_p|\tau, R_p)$ is evaluated for the new and old trees, and compared with:

$$\alpha(\mathcal{T}_p, \mathcal{T}_p^\star) = \min\left\{1, \frac{p(\mathcal{T}_p^\star|\mathbf{X}, \mathbf{y})q(\mathcal{T}_p^\star \to \mathcal{T}_p)}{p(\mathcal{T}_p|\mathbf{X}, \mathbf{y})q(\mathcal{T}_p \to \mathcal{T}_p^\star)}\right\}, \tag{2.9}$$

where $q(\mathcal{T}_p \to \mathcal{T}_p^\star)$ denotes the transition probability from tree $\mathcal{T}_p$ (the previous tree) to $\mathcal{T}_p^\star$, which is the proposed new tree (vice-versa for $q(\mathcal{T}_p^\star \to \mathcal{T}_p)$). More specifically, for a GROW move we have:

$$\begin{aligned} q(\mathcal{T}_p \to \mathcal{T}_p^\star) &= \frac{\mathbb{P}(\text{grow})}{b}, \\ q(\mathcal{T}^\star \to \mathcal{T}) &= \frac{\mathbb{P}(\text{prune})}{w^\star}, \end{aligned} \tag{2.10}$$

where $\mathbb{P}(\text{grow})$ is pre-defined (either as uniform between all available moves, or given e.g. a Dirichlet prior), b is the number of terminal nodes in $\mathcal{T}_p$, from which we can grow a new node. Lastly, $w^\star$ and w are the numbers of internal nodes which are parents of two terminal nodes in $\mathcal{T}_p^\star$ and $\mathcal{T}_p$, respectively (Kapelner and Bleich, 2016). Analogously, for a PRUNE move, we have:

$$\begin{aligned} q(\mathcal{T}_p \rightarrow \mathcal{T}_p^\star) &= \frac{\mathbb{P}(\text{prune})}{w}, \\ q(\mathcal{T}^\star \rightarrow \mathcal{T}) &= \frac{\mathbb{P}(\text{grow})}{b-1}, \end{aligned} \tag{2.11}$$

where b and w are as defined before. In a CHANGE or SWAP moves, the transition kernels cancel out as the ratio of the $q(\cdot)$ is always 1 (Chipman et al., 1998). Finally, the full BART algorithm is described in Algorithm 1.

Algorithm 1 BART algorithm

1: **Input**: $\mathbf{y}$, $\mathbf{X}$, and number of trees P.
2: **Start**: $\{\mathcal{T}_t\}_1^T$ and set all hyperparameters of the prior distributions.
3: **for** ($p = 1$ to P) **do**
4: Compute $\mathbf{R}_p = \mathbf{y} - \sum_{j \neq p}^{P} G\left(\mathbf{X}; \mathcal{T}_j, \Theta_j\right)$.
5: Propose a new tree $\mathcal{T}_p^\star$ via a GROW, PRUNE, CHANGE, or SWAP move.
6: Compare the current $(\mathcal{T}_p)$ with the proposed tree $(\mathcal{T}_p^\star)$ trees via a Metropolis-Hastings step:
$$\alpha\left(\mathcal{T}_p, \mathcal{T}_p^\star\right) = \min\left\{1, \frac{p\left(\mathcal{T}_p^\star \mid \mathbf{R}_p, \sigma^2\right) q\left(\mathcal{T}_p^\star \rightarrow \mathcal{T}_p\right)}{p\left(\mathcal{T}_p \mid \mathbf{R}_p, \sigma^2\right) q\left(\mathcal{T}_p \rightarrow \mathcal{T}_p^\star\right)}\right\}.$$
7: Sample $u \sim \text{Uniform}\,(0, 1)$:
If $\alpha\left(\mathcal{T}_p, \mathcal{T}_p^\star\right) > u$, set $\mathcal{T}_p = \mathcal{T}_p^\star$.
8: Update all terminal node parameters $\mu_{p,b}$ via $p(\mu_{p,b}|\mathbf{R}_p, \mathcal{T}_p, \tau)$, for $b = 1, \ldots, b_p$.
9: **end for**
10: Update τ via $p(\tau|\mathcal{T}, \boldsymbol{\Theta}, \mathbf{y})$.
11: Update $\hat{\mathbf{y}} = \sum_{p=1}^{P} g\left(\mathbf{X}, \mathcal{T}_p, \Theta_p\right)$.

Lastly, it is worth discussing BART for a classification case, where the outcome variable (or target variable) can be either binary or multiclass (Kindo et al., 2016b; Murray, 2021). For the binary case, the difference is that we introduce a latent

variable $z_i \sim N(\sum_{p=1}^{P} G(\mathbf{x}_i; \mathcal{T}_p, \Theta_p), 1)$, $i = 1, \cdots, n$ where $y_i = 1$ if $z_i > 0$ and $y_i = 0$ otherwise, since the Normal distribution is symmetric. This means that we use the Normal latent variable as a way of mapping its continuous values to the binary space, and with that, modelling a class feature with BART. This model is based on the standard normal cumulative distribution function, as we can use it to create $P(y_i = 1|\mathbf{x}_i) = \Phi\left(\sum_{p=1}^{P} G\left(\mathbf{X}, \mathcal{T}_p, \Theta_p, \right)\right)$, and the precision parameter is more often set to 1. In this case the cumulative distribution function works like a link function, mapping the values estimated by $G(\cdot)$ to the binary support. As for the algorithm, a few changes are required for this model to work, such as defining priors on the center parameters μ that assign high probabilities to the interval $\Phi(-3)$ and $\Phi(3)$, since we are using the Normal distribution. The partial residuals become based on z_i instead of y_i, and are defined as $\mathbf{R}_t = \mathbf{z} - \sum_{j \neq p}^{P} G\left(\mathbf{X}, \mathcal{T}_j, \Theta_j\right)$.

2.4 Conclusions

Tree-based machine learning algorithms provide a powerful and intuitive framework for tackling complex prediction tasks, balancing interpretability and predictive performance. Methods such as decision trees, random forests, and BART the inherent flexibility of trees, enabling them to capture nonlinear relationships and interactions within the data. Bayesian Additive Regression Trees (BART) extend the overall concept of tree-based methods by incorporating a Bayesian framework, offering robust uncertainty quantification and improved prediction accuracy. BART combines the strengths of tree-based methods with probabilistic modeling, making it a valuable addition to the machine learning toolkit for both practitioners and researchers seeking to balance interpretability, accuracy, and uncertainty estimation in predictive modeling. The remainder of this thesis explores several extensions of BART to more complex scenarios, enabling this powerful algorithm to tackle a wider range of problems. These enhancements aim to expand BART's applicability, making it a versatile tool for modeling and solving diverse challenges in various domains.

CHAPTER 3

Generalizing Gain Penalization for Feature Selection in Tree-based Models

3.1 Introduction

In many Machine Learning problems, features can be hard or economically expensive to obtain, and some may be irrelevant or poorly linked to the target. For these reasons, reducing the number of features is an important task when building a model, and benefits the data visualization and model performance, whilst reducing storage and training time requirements (Guyon and Elisseeff, 2003). However, for tree-based methods, there is no standard procedure for feature selection or regularization in the literature, as one would find for Linear Regression and the LASSO (Tibshirani, 1991) for example. Performing feature selection in trees can be difficult, as they struggle to detect highly correlated features and their feature importance measures are not fully trustworthy (Louppe, 2014). Several methods to tackle this problem have been recently proposed, including (Diaz-Uriarte, 2007), (Friedman and Popescu, 2008), and (Deng and Runger, 2012).

In (Friedman and Popescu, 2008), the authors treat trees as parametric models and use procedures analogous to LASSO-type shrinkage methods, by penalizing

the coefficients of the base learners and reducing the redundancy in each path from the root node to a leaf node. However, their selected features can still be redundant, since the focus is on reducing the number of rules instead of the number of features.

On a different approach we have (Diaz-Uriarte, 2007), which focuses on gene selection specifically for classification methods. The authors propose an iterative tool that eliminates the least important features (in fractions of the number of features, p) and updates the algorithm at each iteration. The complication is that the method will always be either computationally expensive, if p is low, or will eliminate too many features at once, which can exclude useful or interaction features. Besides, the method does not generalize to other dataset contexts or tasks, such as regression.

In the contrasting approach of (Deng and Runger, 2012) and (Deng and Runger, 2013), the authors regularize Random Forests by gain penalization. Their method consists of letting the features only be picked by a Random Forest if their penalized (weighted) gain is still high. They make recommendations on how to set the penalization coefficients and present their implementation in the `RRF` package for `R` (R Core Team, 2018). However, the authors give no further guidelines on how to generalize their method for other tree-based models and penalization types and do not explore the influence of hyperparameters on the algorithm.

Given that, in this work, we develop a gain penalization approach that is fully generalizable and widely applicable, in opposition to those mentioned above. In particular, our main contributions are:

- We provide a general gain penalization procedure for tree-based models, which allows for a combination of local and global regularization parameters.
- We allow for the bespoke local regularization functions to be domain-specific, which introduces a prior-like component to feature selection in tree-based models.
- We propose different techniques for setting the regularization parameters and discuss how they affect the final results, with real and simulated examples.

- We generalize gain penalization to multiple tree-based methods (CART, Bagging, Random Forests), for both regression and classification.
- We make available a faster implementation of the gain penalization method, included in the very widely used `ranger` package.

The format of this paper is structured as follows. Section 3.2 explains the problem setup, followed by the generalization of gain penalization in Section 3.3. In Section 3.4, we present the results for simulated and real data. Section 3.5 explains the implementation details, and Section 3.6 has the conclusions and future work.

3.2 Problem setup

Consider a set of training target-feature pairs $(Y_i, \mathbf{x}_i) \in \mathbb{R} \times \mathbb{R}^p$, with $i = 1, \ldots, N$ indexing the observations with p being the total number of features. In general, we can estimate an $\hat{f}$ that describes how the features $\mathbf{x}_i$ relate to Y_i and use it for prediction or inference. However, not all features need to be involved in $\hat{f}$. Especially for tree-based models, the occurrence of noisy or correlated features can badly influence the results (Strobl et al., 2008). Given that, our interest here relies on estimating $\hat{f}$ such that it will only use the matrix $\mathbf{x}_A$, composed by the sub-vectors of $\mathbf{x} \in \mathbb{R}^p$ indexed by A, $A \subset \{1, \ldots p\}$, which should contain the optimal set of features (it produces similar or equal prediction errors as the full set of features), that is potentially of a much smaller dimension.

3.2.1 Trees

Non-linear models have been extensively used for regression and classification problems. Trees are a particular case of such models, that recursively partition the feature space, resulting in a local model for each estimated region (Breiman et al., 1984). They learn the features directly from the training data, creating an adaptive basis function model (ABM) (Murphy, 2012) of the form

$$f(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] = \sum_{r=1}^{R} w_r \mathbb{I}(\mathbf{x} \in R_r) = \sum_{r=1}^{R} w_r \phi_r(\mathbf{x}; \mathbf{v}_r), \tag{3.1}$$

where R_r is the r-th region, w_r is the prediction given to this region and $\mathbf{v}_r$ represents the splitting feature chosen and the corresponding splitting value. These algorithms are fitted using a greedy procedure, that computes a locally optimal maximum likelihood estimator by finding the splits that lead to the minimization of a cost function. For regression, the cost function of a decision $\mathbb{D}$ is frequently defined as $cost(\mathbb{D}) = \sum_{i\in\mathbb{D}}(y_i - \bar{y})^2$, where $\bar{y} = (\sum_{i\in\mathbb{D}} y_i)|\mathbb{D}|^{-1}$ is the mean of the observations in the specified region, while for classification this function is replaced by the misclassification rate, or $cost(\mathbb{D}) = |\mathbb{D}|^{-1}\sum_{i\in\mathbb{D}}\mathbb{I}(y_i \neq \hat{y})$.

Since trees are in general considered non-probabilistic algorithms, one way of measuring the importance of each feature is to calculate and aggregate their split gains. The gain of a new split is a normalized measure of the cost reduction, given by

$$\begin{aligned}\Delta(i,t) =& cost(\mathbb{D}) - \\ &\left(\frac{|\mathbb{D}_{LN_{(i,t)}}|}{|\mathbb{D}|}cost(\mathbb{D}_{LN_{(i,t)}}) + \frac{|\mathbb{D}_{RN_{(i,t)}}|}{|\mathbb{D}|}cost(\mathbb{D}_{RN_{(i,t)}})\right),\end{aligned} \tag{3.2}$$

for feature i at splitting point t, while $\mathbb{D}$ is related to the previous estimated split, $LN =$ (left candidate node) and $RN =$ (right candidate node). The global importance value is given by accumulating the gain over a feature, $\Delta(i) = \sum_{t\in\mathbb{S}_i}\Delta(i,t)$, where $\mathbb{S}_i$ now represents all the splitting points used in a tree for the $i-$th feature. This measure will be a key component in our developments below.

3.2.2 Tree Ensembles

Trees are known to be high variance estimators: small changes in the data can lead to the estimation of a completely different tree (Murphy, 2012). One way to increase stability is to use the property that an average of many estimates has a smaller variance than one estimate, and grow many trees from re-samples of the data. Averaging such results give us a bagged ensemble (Breiman, 1996b) of the form

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N_{tree}} \frac{1}{N_{tree}}\hat{f}_n(\mathbf{x}), \tag{3.3}$$

where $\hat{f}_n$ corresponds to the n-th tree. The Random Forest (Breiman, 2001b) algorithm, for example, is defined by allowing only a random subset m of the features to be the candidates in each split. As for the importance values, in tree ensembles the feature importances get averaged over all the trees, or

$$Imp_i = \frac{1}{N_{tree}} \sum_{n=1}^{N_{tree}} \Delta(i)_n, \tag{3.4}$$

where i represents the feature index. Moreover, the prediction performance of the trees in ensembles such as the Random Forest relies on the number of features tried at each split, called `mtry` here, as when `mtry` $\to 1$, the larger the variance of each tree, but the more effective will be the averaging process, and vice versa (Louppe, 2014). As we will see later, the gain penalization procedure also depends on `mtry`, because when the wrong value is set for this hyperparameter, the penalization can lead to models that are very far from the truth.

3.2.3 Regularization by gain penalization

In (Deng and Runger, 2012), the authors first discuss the regularization of Random Forests by gain penalization. The *Regularized Random Forest* (*RRF*) proposes weighting the gains of the splits during the greedy procedure, guiding the feature choosing of the model. The regularized gain is defined as

$$\text{Gain}_R(\mathbf{X}_i, t) = \begin{cases} \lambda\Delta(i,t),\ i \notin \mathbb{U} \text{ and} \\ \Delta(i,t),\ i \in \mathbb{U}, \end{cases} \tag{3.5}$$

where $\mathbb{U}$ is the set of indices of the features previously used, $\mathbf{X}_i$ is the candidate feature, and t the candidate splitting point. The $\lambda \in (0, 1]$ hyperparameter is the penalty coefficient that controls the amount of regularization each feature receives. A feature is penalized if it is new to the whole ensemble, as the method has a memory of which features were already used. Naturally, λ can be a constant value for all the features but ideally, there should be a regularization parameter for each feature that best represents the information they carry about the target. In (Deng and Runger, 2013), the authors modify this idea by introducing the

Guided RRF. It consists of first running a Standard Random Forest (`mtry` $\approx \sqrt{p}$, number of trees = 500) and producing an importance measure for each feature and scaling this measure, in order to find $Imp_i^{'} = \frac{Imp_i}{max_{j=1}^{P} Imp_j}$ where Imp_i is the importance measure calculated for the i-th feature in the Random Forest. The $Imp_i^{'}$ becomes a component of the penalization factor, in the form $\lambda_i = (1 - \gamma) + \gamma Imp_i^{'}$. However, this method is explicitly developed for Random Forests, as the gain penalization itself depends on the results of a previously run Random Forest. No other methodologies or extensions are explored, and the influence of the Random Forests hyperparameters are not studied by the authors.

3.3 Generalizing Gain Penalization

One of our goals with this work is to propose a generalization of gain penalization in tree-based models for feature selection. This generalization is made in two senses: for the penalization methodology and in the algorithm type. For the algorithm, this means that the regularization method can be applied to any tree-based algorithm, be it either single trees such as CART, or elaborated ensembles like Bagging and Random Forests. All these methods are available in our implementation. As for the penalization method, we generalize it by proposing a gain penalization based on a λ_i parameter, that is written as

$$\lambda_i = (1 - \gamma)\lambda_0 + \gamma g(\mathbf{x}_i), \tag{3.6}$$

where $\lambda_0 \in [0, 1)$ is interpreted as the baseline regularization, $g(\mathbf{x}_i)$ is a function of the i-th feature, and $\gamma \in [0, 1)$ is their mixture parameter, with $\lambda_i \in [0, 1)$. In this fashion, we propose a local-global form of penalization, that is applied to all the features used in the model. The equation balances how much all features should be jointly, or globally, penalized and how much will it be due to a local $g(\mathbf{x}_i)$, that is manually set. When $\gamma = 0$, only a global penalization is performed, while when $\gamma = 1$, the regularization is fully controlled by $g(\mathbf{x}_i)$.

The $g(\mathbf{x}_i)$ should represent relevant information about the features, based on some characteristic of interest. It can include, for example, external information about the relationship between $\mathbf{x}_i$ and $\mathbf{y}$, or information only about $\mathbf{x}_i$ and its utility for

the model. This formulation has inspiration on the use of priors made in Bayesian methods since we intend to introduce prior knowledge regarding the importance of each feature into the model. In the same way, the data will tell us how strong our assumptions about the penalization are, since even if we try to penalize a truly important feature, its gain will be high enough to overcome the penalization and the feature will get selected by the algorithm.

3.3.1 Choosing $g(\mathbf{x}_i)$

In the following, we list a few $g(\mathbf{x}_i)$ options, taking into consideration the possible natures of the features.

Correlation: A familiar option for continuous features is just to use $g(\mathbf{x}_i)$ as the absolute value of the marginal correlation between $\mathbf{x}_i$ and $\mathbf{y}$, when we assume a continuous target problem. It could be either Pearson's, Kendall's, Spearman's, or any other correlation coefficient of preference (the first is more suitable for ordinary numeric inputs, while the others will be more convenient for ordered inputs (Chen and Popovich, 2002)). We drop the *sign* because when two features are correlated, the magnitude of the coefficient is enough to define its importance in terms of significance, and one can use simply

$$g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|. \tag{3.7}$$

Entropy and Mutual Information: A different situation is when the features are discrete. In information theory, Shannon's entropy (Shannon, 1948) is a measure of the uncertainty of a (discrete) random feature. In a short description, if a discrete feature X has K states, its entropy will be calculated as

$$\mathbb{H}(X) = -\sum_{k=1}^{K} p(X = k) \log_2 p(X = k), \tag{3.8}$$

where $\mathbb{H}(X) \in [0, \infty]$. Higher entropy will mean more uncertainty, so it can be reasonable to give more weight to features with lower uncertainties. One can use a normalized version of the entropy calculated for each $\mathbf{x}_i$, or

$$g(\mathbf{x}_i) = 1 - \frac{\mathbb{H}(\mathbf{x}_i)}{max_{j=1}^{P}\mathbb{H}(\mathbf{x}_j)}, \tag{3.9}$$

compelling the features with lower entropy to have larger penalization coefficients. Under the same framework, a more general approach to quantify how much knowing about one feature tells us about the other is the Mutual Information function. In this case, the similarity between a joint distribution $p(X, Y)$ and a factored distribution $p(X)p(Y)$ is calculated with

$$\text{MutInf}(X; Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{3.10}$$

for two features X and Y, which is basically the Kullback-Leibler divergence between the two distributions (Murphy, 2012). Recalling Equation 3.8, it is easy to see that the Mutual Information value is the reduction in the uncertainty about Y when we observe X, so it can be straightforwardly used as

$$g(\mathbf{x}_i) = \frac{\text{MutInf}(\mathbf{x}_i, \mathbf{y})}{max_{j=1}^{P}\text{MutInf}(\mathbf{x}_j, \mathbf{y})}. \tag{3.11}$$

Boosted: If there is no interest in differentiating continuous or discrete features, one can use what we call a *Boosted* $g(\mathbf{x}_i)$. Such functions depend on previously run machine learning models that provide an importance value for the features. The term *Boosted* is to introduce some familiarity with what the algorithm consists of since we can arguably see it as a heterogenous Boosting (Nascimento and Coelho, 2009) applied to the features instead of the observations.

More generally, many Machine Learning algorithms can be used whenever they allow for the calculation of an importance value. Some examples include: Generalized Linear Models (Nelder and Wedderburn, 1972), where e.g. the normalized absolute parameter coefficients can be interpreted as importance values, and Support Vector Machines (Hastie, Trevor, Tibshirani, Robert, Friedman, 2009) that produce importance values via sensitivity analysis ((Cortez and Embrechts, 2013; Cortez, 2016)). Each family of algorithms will have its specific characteristics and

preferences towards the features, so it is advisable to be aware of those details when using it in a *Boosted* $g(\mathbf{x}_i)$.

Combination: Another possibility is combining two or more $g(\mathbf{x}_i)$. Objectively speaking, some functions will be more appropriate to one type of feature than others. As an example, one could combine a *Boosted* method with the marginal correlations between the target and each feature. This formulation can, for example, be written as

$$g(\mathbf{x}_i) = \begin{cases} |corr(X_i, y)|, \textbf{ if } |corr(X_i, y)| > \epsilon \\ (Imp_i^{'}, \textbf{ if } |corr(X_i, y)| \leq \epsilon \end{cases}, \tag{3.12}$$

where we let the absolute values of the correlations compose $g(\mathbf{x}_i)$ if the correlation is over a certain threshold ϵ, and use $Imp_i^{'}$ from a previously run algorithm. Analogously, we might want to explore the features that are not so correlated to the target (e.g. when their relationship is very non-linear) by using a high value for ϵ.

3.3.2 Depth parameter

Sometimes, growing very bushy trees with new features is not desirable when we want to use the smallest set of features possible. Following (Chipman et al., 2010a), where the authors use prior distributions for whether a new feature should be picked in a Bayesian Regression Tree, we introduce the idea of increasing a penalization given the current depth of the tree. Their priors take into account the current depth of a tree, so when a tree is already deep the priors get less concentrated in high probability regions, resulting in lesser bushier trees. In our framework, a similar idea is applied by setting

$$Gain_R(\mathbf{X}_i, t, \mathbb{T}) = \begin{cases} \lambda_i^{d_\mathbb{T}} \Delta(i,t), \; i \notin \mathbb{U} \text{ and} \\ \Delta(i,t), \; i \in \mathbb{U}, \end{cases} \tag{3.13}$$

where $d_\mathbb{T}$ is the current depth of the $\mathbb{T}$ tree, $\mathbb{T} = (1, \ldots, \texttt{ntree})$, for the i-th feature. The aim here is to reduce the gains of the features if they are to be picked in a deep

node, preventing new features to appear at the bottom of trees unless their gains are exceptionally high. The benefit of this comes from the fact that deep nodes contain fewer observations than their parents, so a deep split will likely lead to a smaller gain if any at all. In a scenario where we want to keep only the variables that have a high importance to the model, this is undesirable and can be prevented by using our method added with the depth penalization.

3.3.3 Details & advantages

Feature masking effect: Tree-based models often suffer from feature masking effects (Louppe, 2014). For example, in a tree, some feature X_j might never occur in the algorithm if it leads to splits slightly worse than some other feature X_i. So if X_i is removed, X_j can prominently occur and have a high importance value. In theory, this problem is overcome by ensembles like Random Forests, as selecting only m features to pick from decorrelates the trees, but if we regularize Random Forests, the problem remains. This happens because if weak features end up being picked (randomly) by the trees, their gains will have an unfair advantage against the other features (possibly very important features), that will be penalized. Luckily this situation is can be fixed with hyperparameter tuning for `mtry`.

Correlated features: A second issue to have in mind when working with tree ensembles is their bias towards giving high importance to correlated features (Strobl et al., 2008). As an example, suppose we have a subset $\mathbf{C} \subseteq \mathbf{X}$ of features which are correlated. Ideally, we would expect to have only one or just a few of these features being selected, because if one of the correlated features is truly important for prediction, using this one feature is enough, but the ensembles are not able to detect and eliminate correlated features. The naive approach to tackle this problem is to calculate a full correlation matrix between all the features and filter by only the least correlated one, but when p grows this might not be computationally feasible, and it implies more manual work when building the model. Our gain penalization method automatically deals with the correlated features, since when one of the features in $\mathbf{C}$ gets picked, the algorithm is less likely to pick the other correlated features as well, given that a new feature needs to reduce the prediction

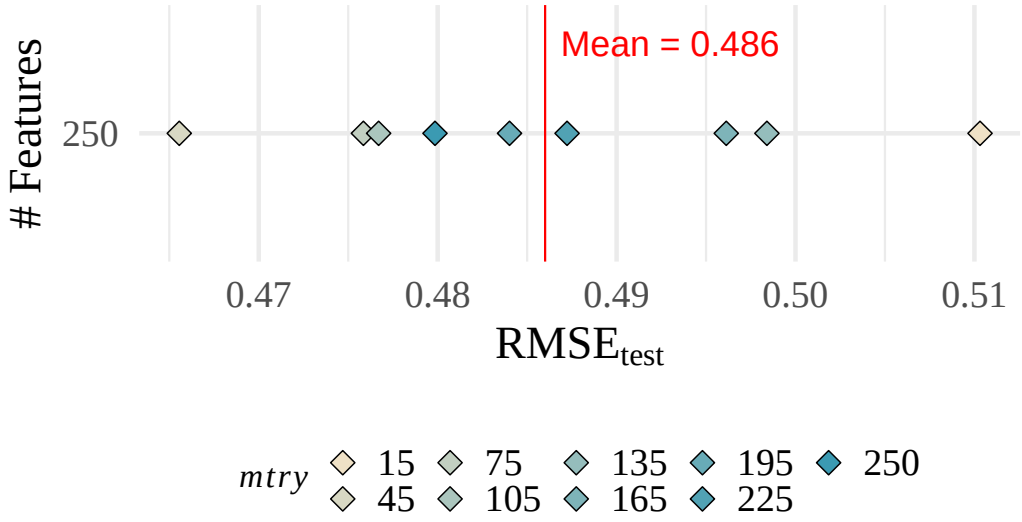


Figure 3.1: Averages of the number of features used and $RMSE_{test}$ values for a Standard Random Forest. The models use all 250 features in every run. The lowest RMSE occurs with `mtry` = 45 and the highest RMSE with mtry = 15.

error more drastically to be selected.

3.4 Experiments

This section shows the results of our experiments, that evaluated the effects of different regularization types in simulated and real datasets using the Random Forest algorithm.

3.4.1 Simulated data

Consider now a set $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{205})$ of features, all sampled from a Uniform[0, 1] distribution, $n = 1000$. We generated a target of interest $\mathbf{Y} \in \mathbb{R}$ as

$$\begin{aligned}\mathbf{y} =& 0.8sin(\mathbf{x}_1\mathbf{x}_2) + 2(\mathbf{x}_3 - 0.5)^2 + 1\mathbf{x}_4 + 0.7\mathbf{x}_5 + \\ & \sum_{j=1}^{200} 0.9^{(j/3)}\mathbf{x}_{j+5} + \sum_{j=1}^{45} 0.9^j\mathbf{x}_5 + \epsilon,\ \epsilon \sim N(0,1),\end{aligned} \tag{3.14}$$

inspired by the simulation equation proposed in (Friedman, 1991), totaling 250 features. This framework produces interesting relationships between the target and the features: non-linearities ($i = (1, 2, 3)$), decreasing importances ($i = (6, \ldots, 205)$) and correlations ($i = (5, 206, \ldots, 250)$), inducing a more complicated scenario. We created 10 datasets, all randomly split into train and test set (80%/20%). For all the algorithms we fixed the number of trees at 100, varied `mtry` $= (15, 45, 75, 105, 135, 165, 195, 225, 250)$ and our accuracy measure is the RMSE calculated in the test set. We used a standardized version of $\mathbf{y}$ and, in the following, the term *selected feature* represents any feature with importance $\Delta(i) > 0$ in the final estimated model.

3.4.2 Standard Random Forest, *RRF* and *GRRF*

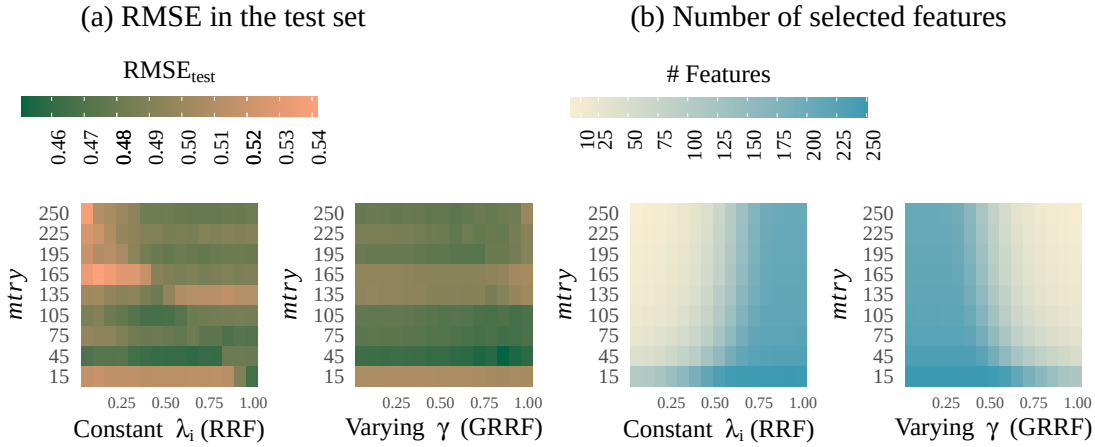


Figure 3.2: (a) Tile plot for the average of resulting $\text{RMSE}_{\text{test}}$ and (b) number of selected features in a *Regularized Random Forest* and a *Guided Regularized Random Forest* varying `mtry`, λ and γ. The number of selected features has a clear effect in the two models. For the *RRF*, the region with the lowest $\text{RMSE}_{\text{test}}$ is predominantly the one with the most features, while for the *GRRF* this situation improves.

As a benchmark, we run a Standard Random Forest, the *RRF* and *GRRF* models for each of the the 10 simulated datasets and all the different values of `mtry`, and the results are compared to our method. The first `mtry` is what would be the default in a Standard RF, since $\sqrt{250} \approx 15$, and the last is the total of features available.

For the Random Forest model, the resulting number of features used for all the models is always the maximum available (see Figure 3.1). If we consider the correlated features issue, this means that too many features are being picked, once we know that they become irrelevant in their joint presence. The $\text{RMSE}_{\text{test}}$ changes when `mtry` changes: when `mtry` $= 45$ is when we have the best results, meaning that the default value (`mtry` $= \sqrt{p} \approx 15$) is not the best option. As for the *RRF* and *GRRF*, we used the hyperparameters $\lambda = (0.05, 0.12, 0.18, 0.25, 0.32, 0.39, 0.45, 0.52, 0.59, 0.65, 0.72, 0.79, 0.86, 0.92, 0.99)$,$\gamma = (0.05, 0.12, 0.18, 0.25, 0.32, 0.39, 0.45, 0.52, 0.59, 0.65, 0.72, 0.79, 0.86, 0.92, 0.99$ (which represents the weighting parameter of $Imp_i^{'}$ in the *GRRF*) and tried all combinations between λ (*RRF*), γ (*GRRF*) and `mtry` (both). The models were run using the `RRF` [1] (Deng, 2013) package for `R` (R Core Team, 2018).

Figure 3.2 shows the results of the average $\text{RMSE}_{\text{test}}$ (left) and average number of selected features (right) in the 10 datasets for the two types of models. We can see a continuous transition in the number of features picked by the two models, but they present an inverse pattern regarding the `mtry` and penalization parameters. For the *RRF*, the region with the lowest $\text{RMSE}_{\text{test}}$ is predominantly the one with the most features, meaning that the penalization is happening but the models using the least features do not satisfactorily predict for the test set. As for the *GRRF*, `mtry` $= 45$ seems to be optimal, similarly to the Random Forest. However, the lowest $\text{RMSE}_{\text{test}}$ region happen when γ is high, which represents and improvement but still seemingly leads to the number of selected features to not be as low as it can be.

3.4.3 Generalized Gain Penalization in Random Forests

Now we present the experiment results using the Generalized Gain Penalization in Random Forests. For this subsection, we vary $\lambda_0 = (0.1, 0.5, 0.9)$ and $\gamma = (0.001, 0.25, 0.5, 0.75, 0.99)$, use all combinations of the hyperparameters $(\gamma \times \lambda_0)$,

[1] There is a difference on the splitting criteria of the `RRF` package, that calculates $\Delta(i,t) = \left(\frac{cost(\mathbb{D}_{LN_{(i,t)}})}{|\mathbb{D}_{LN_{(i,t)}}|} + \frac{cost(\mathbb{D}_{RN_{(i,t)}})}{|\mathbb{D}_{RN_{(i,t)}}|}\right) - \frac{cost(\mathbb{D})}{|\mathbb{D}|}$, when finding the best feature and threshold to split on. At each time the cost reduces in $\frac{cost(\mathbb{D})}{|\mathbb{D}|}$, making the cost measure have a smaller magnitude than the one used by us in the `ranger` package.

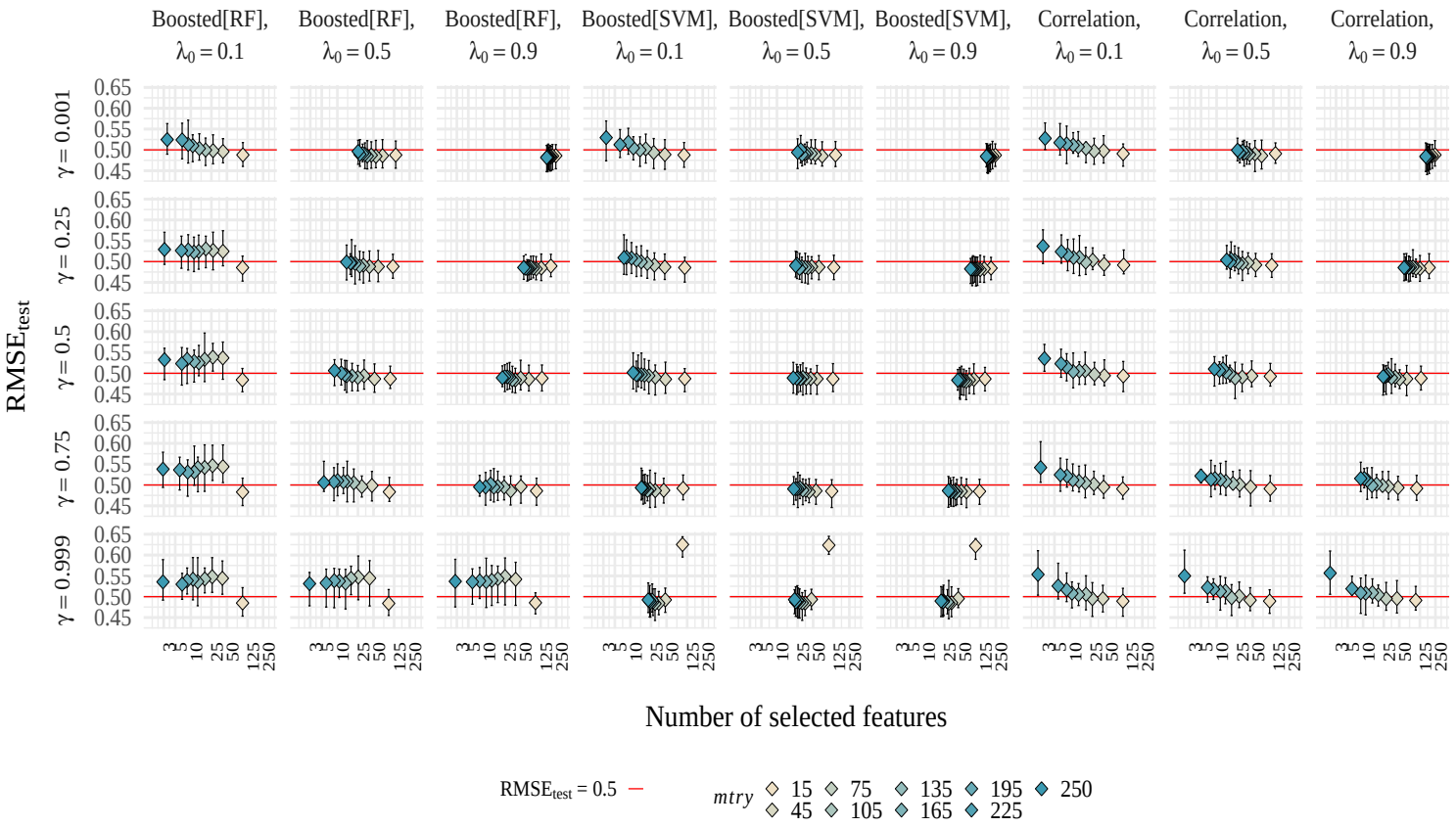


Figure 3.3: Averages of $RMSE_{test}$ (with maximum and minimum intervals) and of the **log** number of features using the mixture of a λ_0 and a $g(\mathbf{x}_i)$, for $g(\mathbf{x}_i) = (|corr(\mathbf{y}, \mathbf{x}_i)|$, Boosted$_{RF}$, Boosted$_{SVM})$. The x-axis shows the original scale, but the values are transformed to log. The models are mainly using fewer features than the *GRRF* or Standard RF, with λ_0 and `mtry` visibly affecting the results.

first with $g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|$ and later using two *Boosted* methods, with a Standard Random Forest and with a Support Vector Machine. In Figure 3.3 we can see that $\text{RMSE}_{\text{test}}$ values are mostly close or below the 0.5 line. In comparison to Figure 3.2, our algorithm is doing better, as we can spot many cases where the $\text{RMSE}_{\text{test}}$ is low while using very few features (<25), especially when $g(\mathbf{x}_i) = Boosted_{SVM}$. Even when the $\text{RMSE}_{\text{test}}$ is a little higher, 0.5 for example, the number of features used is frequently much smaller (<10) than previously seem in the other algorithms. When γ is low the regularization is primarily controlled by λ_0, and we spot a heavier influence of `mtry` on the number of selected features, which tends to decrease as λ_0 increases. When γ is high the penalization values depend more on $g(\mathbf{x}_i)$, and the results vary less regarding the values for λ_0 and `mtry`.

Table 3.1: Percentages of the most important and of correlated features selected and $\text{RMSE}_{\text{test}}$, averaged by `mtry` and γ. When using $g(\mathbf{x}_i) = \text{Boosted}_{SVM}$, we pick more of the important features, less of the correlated and have lower $\text{RMSE}_{\text{test}}$. The *GRRF* tends to pick many of the correlated features, leading to non-optimal feature subsets.

$g(\mathbf{x}_i)$	λ_0	% Imp.	% Corr.	$RMSE_{test}$
Correlation	0.10	65.2%	18.8%	0.51
Correlation	0.50	64.6%	19.0%	0.50
Correlation	0.90	64.6%	20.0%	0.49
Boosted_{RF}	0.10	69%	33.2%	0.52
Boosted_{RF}	0.50	71.2%	28.2%	0.50
Boosted_{RF}	0.90	67.6%	28.0%	0.50
Boosted_{SVM}	0.10	71%	21.6%	0.50
Boosted_{SVM}	0.50	69.8%	20.8%	0.49
Boosted_{SVM}	0.90	67.6%	22.4%	0.48

GRRF

γ	% Imp.	% Corr.	$\text{RMSE}_{\text{test}}$
≈ 0.10	63.8%	33.1%	0.48
≈ 0.50	67.6%	32.6%	0.48
≈ 0.90	82.8%	32.0%	0.48

A more in-depth analysis of the selected features can be seen in Table 3.1. We define the importance of features in the simulation as $\mathcal{V} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \underset{i \in [6,205] \cap [0.9^{(i-5)/3} > 0.01]}{\mathbf{x_i}})$. We can understand this formula as way of decreasing the importance of the features, as when the index grows, their importance tends to get smaller by definition. Also, we do not include the last 45 features which are correlated and we ideally want to avoid them. We then calculate the percentage of important features that were selected by each algorithm, from the total of features, and for the correlated ones, which percentage of those was selected by the algorithm. So, for example, if an algorithm picked 10 features, 3 of them being important, 5 being from the correlated group and 2 being "non-important", we calculate the proportion of important features as 3/5 and the proportion of correlated as 5/45.

With Table 3.1 we see that the proportion of important features is considerably higher for our approach with $g(\mathbf{x}_i) = Boosted_{RF}$ and $g(\mathbf{x}_i) = Boosted_{SVM}$. When we use $g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|$ the algorithm picks less of the correlated features. We also notice that the best results happened when $\lambda_0 \leq 0.5$ and $g(\mathbf{x}_i)$ has a higher influence in the penalization coefficients, so the introduction of prior information in the gain penalization is really helping the feature selection. We also show the same results for the *GRRF*, which picks many more of the correlated features and, on one occasion, more of the important ones. When looking at this result, we need to take into account that this model also tends to select more features in general, so that is not satisfactory if too many variables were selected.

3.4.4 Real Data Classification

This part of our work now discusses the results for real gene classification microarray datasets, specified in detail in Table 3.2. With an average of 4787 features, 67 observations, and 3 classes, those are classical examples of "large p, small n" datasets. Though the focus here is in gene datasets, our method generalizes to data from any contexts or sizes and the datasets were chosen following previous literature ((Diaz-Uriarte, 2007), (Deng and Runger, 2012)).

As the goal here is to find the best features to predict the gene classes, the ex-

Table 3.2: Real classification datasets and its specifications. Problematic $p > n$ situation in all cases.

Dataset	Ref.	Obs.	Features	Classes
adenocarcinoma	[(Ramaswamy et al., 2003)]	76	9869	2
brain	[(Pomeroy et al., 2002)]	42	5598	5
breast 2	[(Van't Veer et al., 2002)]	77	4870	2
breast 3	[(Van't Veer et al., 2002)]	95	4870	3
colon	[(Alon et al., 1999)]	62	2001	2
leukemia	[(Golub et al., 1999)]	38	3052	2
lymphoma	[(Alizadeh et al., 2000)]	62	4027	3
nci 60	[(Ross et al., 2000)]	61	5245	8
prostate	[(Singh et al., 2002)]	102	6034	2
srbct	[(Khan et al., 2001)]	63	2309	4

periment conducted for this section is different. We run the penalized models and extract their selected features, that are later used in a Standard Random Forest, with which the misclassification rates are calculated. This is to mimic how such an approach can be used in practice, where first a discovery experiment is run to identify important features, then a subsequent algorithm is run on a new dataset using the selected features to better assess their prediction power. We set $\gamma = \lambda_0 = 0.5$, attributing the same weight to the baseline regularization and to $g(\mathbf{x}_i)$. We vary `mtry` $= (\sqrt{p}, 0.15p, 0.40p, 0.75p, 0.95p)$ and $g(\mathbf{x}_i) = \left(\text{Boosted}_{RF}, \frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y},\mathbf{x}_j)}\right)$. We also run a Standard Random Forest, a *GRRF*, the LASSO (Tibshirani, 1991), and varSelRF (Diaz-Uriarte, 2007) algorithms for each dataset, which are namely the biggest competitors of our method and are quite different feature selection techniques, bringing variety to our comparisons. All datasets were randomly separated into 50 different train (2/3) and test sets (1/3). We first find the average misclassification rates (MR) and the number of features used for each of the 50 resamples, eliminating at this step the `mtry` column. Out of that, we filter by the resample with the smallest misclassification rate.

According to Table 3.3, the Standard RF uses more features, but does not always have the lowest prediction errors. As for the generalized gain penalization, using

Table 3.3: Average percentage of features used and average misclassification rates with standard deviation for all the models. The gain penalized models used far fewer features than a Standard Random Forest, and it frequently uses fewer variables than the other feature selection techniques. Our approach also frequently has the lowest prediction errors, showing how it can use far fewer features whilst maintaining competitive misclassification performance.

Percentage of features used

Dataset	Std.RF	GRRF	Boosted$_{RF}$	Mut.Inf.	LASSO	varSelRF
adenocarcinoma	9.38	0.83	0.86	0.07	**0.02**	0.05
brain	25.06	1.44	1.46	**0.14**	0.39	0.73
breast 2	24.44	1.76	1.79	**0.14**	0.21	0.34
breast 3	38.58	1.95	2.00	**0.28**	0.67	0.28
colon	34.44	2.53	2.60	**0.44**	0.46	0.94
leukemia	8.07	1.26	1.25	**0.05**	0.28	0.09
lymphoma	13.88	1.07	1.14	**0.08**	0.34	0.72
nci	44.77	1.81	1.88	**0.16**	1.11	0.97
prostate	18.22	1.46	1.40	0.09	0.14	**0.07**
srbct	29.69	2.26	2.25	**0.3**	0.68	0.99

Misclassification rate in the test set

Dataset	Std.RF	GRRF	Boosted$_{RF}$	Mut.Inf.	LASSO	varSelRF
adenocarcinoma	3.45 (0.0)	11.03 (12)	**3.4** (0.0)	10.77 (1.7)	13.83 (6.4)	19.6 (7.7)
brain	8.00 (5.6)	15.38 (14.4)	15.00 (7)	**13.33 (10.5)**	27.6 (11.7)	29 (16.2)
breast 2	25.6 (5.4)	20.7 (2)	**17.8 (7.2)**	20.7 (7.7)	31.56 (5.19)	36.7 (9.1)
breast 3	27.6 (6.4)	30.6 (4)	**28.1 (3.8)**	29.3 (1.6)	29.7 (4.85)	33.9 (8.6)
colon	4.76 (0.0)	**5.71 (2.1)**	6.67 (0.0)	7.78 (3)	16.7 (8.6)	23.06 (8.3)
leukemia	0.0 (0.0)	**0.0 (0.0)**	**0.0 (0.0)**	**0.0 (0.0)**	10.07 (9.4)	13.2 (12)
lymphoma	0.0 (0.0)	**0.0 (0.0)**	**0.0 (0.0)**	**0.0 (0.0)**	1.48 (4.7)	5.86 (4.8)
nci	30.77 (9.4)	38.95 (8)	**37.5 (7.6)**	43.75 (0.0)	42.2 (7.5)	44.7 (12.8)
prostate	0.54 (1.2)	**0.54 (1.2)**	1.08 (1.5)	**0.54 (1.2)**	6.00 (2.8)	8.87 (1.9)
srbct	0.0 (0.0)	**0.0 (0.0)**	0.91 (2)	1.74 (3.9)	1.33 (2.9)	4.5 (3.6)

$g(\mathbf{x}_i) = \left(\frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y})}\right)$ is better for [brain] and [prostate], while when $g(\mathbf{x}_i) =$ Boosted$_{RF}$, the results are good for the [adenocarcinoma], [breast 2], [breast 3] and [nci 60]. The *GRRF* is strictly better for the [colon] and [srbct] datasets considering the MR, though it uses many more features in comparison to the other algorithms. The LASSO often presents a low percentage of features, but

its misclassification rates are much above the ones for the other models shown in the table. If we compare that to the generalized gain penalization method with $g(\mathbf{x}_i) = \left(\frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y})}\right)$, our approach would be much more preferable since it uses less variables than the LASSO while mostly keeping a lower misclassification rate in the test set, The MRs are all the same for the [leukemia] and [lymphona] in the Standard RF, *GRRF* and generalized gain penalization models, but the percentage of features is often the lowest for our method. When this happens and such algorithms also have a low or very similar MR to a Standard RF one, we reach an optimal situation, which happened for almost all the datasets. Looking at the varSelRF results, we notice that this model produces the highest prediction errors and does not beat our models in terms of the percentage of features used, even though this algorithm is designed to work in well in this specific context.

3.5 Implementation

The implementation used here is included as an extension to the `ranger` package (Wright and Ziegler, 2017) for `R` (R Core Team, 2018). This choice was made given that the `ranger`, originally written in `C++`, is the fastest Random Forest implementation available for `R`, so it serves us well for a general tree-based approach and the models can scale to high-dimensional settings. Furthermore, the package has a wide variety of other Random Forests extensions, is actively maintained and interfaces with `python`. The speed and scalability discussion presented in ranger and its comparison to the `randomForest` package (Liaw and Wiener, 2002) is analogous to the one about our regularization implemented in the `ranger` and the one in the `RRF` package (which is based on the original `randomForest` code) so we will not repeat the same experiments[2].

A practical example demonstrating the package extension is available at the URL *https://brunaw.com/blog/posts/2021-03-30-rf-penalization/* , where a gain-penalization analysis is presented, complete with fully reproducible code. Additionally, the code corresponding to the results shown in this document can be accessed on `GitHub` at the repository *https://github.com/brunaw/regularization-rf-paper*. These resources

[2] *All the code and data used are available for reproducing the experiments.*

provide an in-depth look at the implementation details, enabling readers to explore and replicate the analyses discussed in this thesis.

3.6 Conclusions

Feature selection and regularization for tree-based methods is not an easy task and is a topic of active research. In this work, we have demonstrated that one efficient and general way of accomplishing such a task is via a generalization of feature gain penalization for tree-based methods. Our method combines previous information about the features with a baseline penalization λ_0, in a fully flexible local-global form of gain penalization. In general, the technique produces good results in terms of the number of features used and prediction error trade-off, outperforming more traditional methods. The performance and other characteristics of the method have been demonstrated with both simulated and real data in extensive experiments. Along with the methodology, we make the implementation available in the `ranger` package for `R`, which can also be used in `python`.

The downside of our approach is the addition of new hyperparameters, and how to choose them well. Future work involves finding theoretical properties of certain gain penalization approaches, parameter optimization (using e.g. (Snoek et al., 2012)), and comparing our approach to other methods with a similar context ((Johnson and Zhang, 2013; Nan and Saligrama, 2017; Nan et al., 2016), for example). We would also like to explore extensions of this work to other popular and powerful tree-based models, such as gradient boosting algorithms ((Chen et al., 2015; Ke et al., 2017)).

CHAPTER 4

Hierarchical Embedded Bayesian Additive Regression Trees

4.1 Introduction

Bayesian Additive Regression Trees (BART; Chipman et al., 2010b) is a commonly-used probabilistic machine learning approach that produces predictions based on sums of regression trees. Like most standard machine learning approaches, however, BART is not mathematically designed to deal with hierarchical structures in the data. For example, certain observations may share common grouping characteristics e.g. repeated measures and grouped data (Gelman and Hill, 2006), where there is an intra-group variance that needs to be accounted for when fitting the model. There are, of course, some algorithm options that can fit statistical methods to grouped data (e.g. Bates et al., 2011), but they are often not flexible or able to adapt to data that has been generated by a complicated underlying structure. It might also happen that certain observations have their grouping information missing (Vallejo et al., 2011), or predictions might be required at different levels of the data hierarchy, which leads to even more complicated scenarios. Occasionally it is of interest to estimate and/or remove the variability associated with the

[2] **Abbreviations:** BART, Bayesian Additive Regression Trees; HEBART, Hierarchical Embedded Bayesian Additive Regression Trees; MH, Metropolis-Hastings; MCMC, Monte Carlo Markov Chain; RMSE, Root Mean Squared Error; LME, Linear Mixed-Effects;

structured component of the data. The user is often left with a difficult model selection task, as they must choose whether the random effects go into an intercept or a slope or an interaction. Due to the complexity of this task, the model space is not usually well-explored. Literature has arisen on the need to explore and check hierarchical models (see e.g. Chapter 24 of Gelman and Hill, 2006).

To address the challenges of modeling grouped structures while effectively incorporating non-linearities, Hierarchical Embedded BART (HEBART) introduces a hierarchical component tailored for complex grouped data. Unlike traditional methods for including random effects, HEBART models the hierarchical structure by embedding it directly within Bayesian Additive Regression Trees, capturing both the grouped dependency and the non-linear interactions within the data (see Section 7 of Chipman et al. (2010b) for comparison with earlier methods). In its simplest form, the hierarchical component of HEBART is represented by a categorical predictor variable that adjusts predictions across all terminal nodes of each tree. This design extends the effects beyond single elements, allowing predictions to be made at multiple hierarchical levels with greater flexibility and capturing data uncertainties. Furthermore, HEBART does not require grouping information to be present for predictions, enabling its use on data where group identifiers are missing or where the associated variability should be excluded, which adds one more level of flexibility to this model (and also of applicability). This makes HEBART particularly well-suited for real-world applications with complex data structures, and we provide a detailed technical overview of this extension in the following section.

Our paper is structured as follows. In Section 4.2 we introduce the BART model mathematically, and discuss the fitting algorithm and some extensions that have already been proposed. In Section 4.3 we outline our new HEBART approach and discuss how this extends BART into a generalized model for hierarchical data structures, whilst retaining the attractive properties and algorithmic efficiency that BART exhibits. In Section 4.4 we demonstrate the performance of the method on simulated and real-world data. Finally, Section 4.5 discusses some of the potential future research areas and drawbacks of our approach. Later on this document, an appendix contains some of the more detailed mathematics behind the fitting algorithm.

4.2 Introduction to Bayesian Additive Regression Trees

4.2.1 The BART model

The initial Bayesian Regression Tree model was first proposed over 20 years ago (Chipman et al., 1998), and consists of an algorithm that fits CART decision trees using Bayesian inference. The same authors extended this method to create the Bayesian Additive Regression Tree (Chipman et al., 2010b) approach, which assumes that the generating system of a continuous random variable $\mathbf{y} = [y_1, \ldots, y_n]$ can be approximated by a sum of regression trees. In a standard regression setting the BART model is usually written as:

$$y_i = \sum_{p=1}^{P} \mathbf{G}(X_i; \mathcal{T}_p, \Theta_p) + \epsilon_i, \ \epsilon_i \sim N(0, \tau^{-1}), \tag{4.1}$$

for observations $i = 1, \ldots, n$, and $p = 1, \ldots, P$ trees with P the total (fixed) number of trees, X_i represents the set of covariates; $\mathcal{T}_p$ is a tree structure, and Θ_p represents a set of terminal node parameters. The function $\mathbf{G}$ returns a terminal node prediction from Θ by passing X_i through the tree T_p. Θ_p consists of a set of $\mu_{p,l}$ parameters for each of the $l = 1, \ldots, L_p$ terminal nodes in tree p. These values provide the tree-level predictions which are summed together to give an overall predicted value. The residual term ϵ is assumed normal with residual precision τ. Figure 5.1 (left panel) shows a standard single tree that a BART model may use.

We write the set of all trees and parameters as $\mathcal{T}$ and Θ respectively. The joint posterior distribution of the trees and all the parameters is then given by:

$$P(\mathcal{T}, \Theta, \tau | \mathbf{X}, \mathbf{y}) \propto \left[\prod_{p=1}^{P} \prod_{b=1}^{L_p} \prod_{i:\mathbf{x}_i \in \mathcal{D}_{p,b}} p(y_i | \mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau) \right] \times \left[\prod_{p=1}^{P} \prod_{b=1}^{L_p} p(\mu_{p,b} | \mathcal{T}_p) p(\mathcal{T}_p) \right] p(\tau), \tag{4.2}$$

where $p(y_i | \mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau)$ is the normally distributed likelihood as defined in Equation 5.1, $\mathcal{D}$ represents the regions in the covariates space (e.g. tree nodes), and b

indexes the terminal nodes in tree p. The term $p(\mu_{p,b}|\mathcal{T}_p)$ is the prior distribution on the terminal node parameters across each terminal node b in each tree p. $p(\mathcal{T}_p)$ is the prior distribution on the tree structure, and $p(\tau)$ is the prior on the residual precision.

The prior distribution on the trees proposed by Chipman et al. (2010b) involves applying a separate term for each node in the tree and considering both the probability of a split as well as the probability of a new splitting variable being chosen. For an entire tree $\mathcal{T}_p$, we have:

$$P(\mathcal{T}_p) \propto \prod_{\eta \in L_{\mathcal{T}}} (1 - P_{SPLIT}(\eta)) \prod_{\eta \in L_{\mathcal{I}}} P_{SPLIT}(\eta) \prod_{\eta \in L_{\mathcal{I}}} P_{RULE}(\rho|\eta).$$

where $L_{\mathcal{T}}$ and $L_{\mathcal{I}}$ represent the sets of terminal and internal nodes, respectively, ρ represents a generic splitting value, and η represents a node in the tree. The probability of a node being non-terminal is given by $P_{SPLIT}(\eta) = \alpha(1 - d_\eta)^{-\beta}$, where d_η denotes the depth of the node η. The recommended values for the hyperparameters are $\alpha \in (0, 1)$ and $\beta > 0$, which control the depth and bushiness of the trees. For the probability of the new splits, $P_{RULE}(\rho|\eta, \mathcal{T}) = \frac{1}{p_{adj}(\eta)} \frac{1}{n_{j.adj}(\eta)}$ where $p_{adj}(\eta)$ represents how many predictors are still available to split on in node η, and $n_{j.adj}(\eta)$ how many values in a given predictor are still available.

The prior distribution for the terminal node and overall parameters in the standard regression case is denoted by $p(\Theta, \mathcal{T})$. This is given a prior with a standard conjugate form:

$$\mu_1, \ldots, \mu_L|\tau_\mu, \mu_\mu, \mathcal{T} \sim \mathcal{N}(\mu_\mu, \tau_\mu^{-1}),$$

where τ_μ^{-1} and μ_μ are chosen such that a high density of this distribution is apportioned to the range $[y_{min}, y_{max}]$ interval, by setting $P\mu_\mu - k\sqrt{P(\tau_\mu^{-1})} = y_{min}$ and $P\mu_\mu + k\sqrt{P(\tau_\mu^{-1})} = y_{max}$, for some value of k. The response y is usually standardized before the model is run which allows for reasonable guesses as to the hyper-parameter values of μ_μ and τ_μ, though these too can be estimable parameters.

The residual precision prior is set as $\tau \sim \text{Gamma}(\nu/2, \gamma\nu/2)$, where γ and ν are fixed. An oft-used tactic is to set the two hyper-parameters such that the BART residual precision is greater than an equivalent precision value from a standard linear regression model applied to the same data with a high probability. We follow the same guidance in our extension to the model as outlined below. For more details on the full BART model and algorithm, please see Chapter 2 of this thesis.

4.3 Hierarchical Embedded Bayesian Additive Regression Trees (HEBART)

Our HEBART approach merges the ideas from traditional Bayesian hierarchical modeling (Gelman and Hill, 2006) and linear mixed effects models (Pinheiro and Bates, 2000) with BART. We allow each tree to have an extra split on each terminal node corresponding, in the simplest version, to a random intercept for each member of a categorical predictor variable z_i which takes values $j = 1, \ldots, J$ according to the group membership of observation i. Thus we introduce intra-group node parameters which we write as $\phi_{b,j}$ as the estimate for group j in terminal node b. We refer to these parameters as the sub-terminal node level. Here, the term 'embedded' is used to represent the inclusion of the grouping variable into the BART model at the terminal node level rather than as a simple addition on the BART mean, as was originally proposed as an extension to BART in Chipman et al. (2010b). The new parameters allow us to have a group-specific prediction for each node, as well as an overall terminal node prediction μ. The flexibility of this structure means that there is no requirement for the user to specify where the random effect is included, for example as an intercept or as a regression slope. With HEBART we can fit Bayesian Additive Regression Trees to any kind of grouped data where there is such a categorical predictor, such as longitudinal data, repeated measures data, multilevel data, and block designs. In addition, having the two levels of predictions is advantageous for scenarios where the group information is not available for all or a subset of the new data. The Bayesian paradigm allows for imputation of any missing groups at any of the terminal nodes. In Figure 5.1 we show a standard BART tree alongside that of our new HEBART trees.

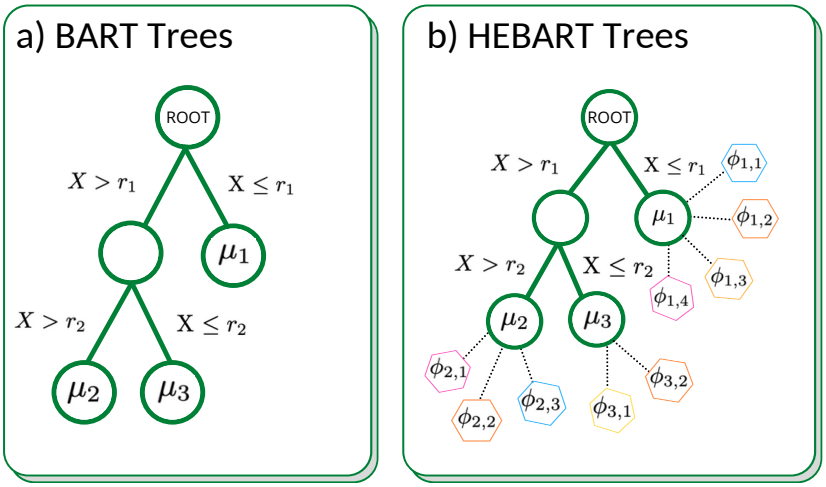


Figure 4.1: Left panel: a standard BART tree using one covariate X. Right panel: a HEBART tree with one covariate X and a single grouping variable with four levels. In BART, each terminal node b has only one terminal node parameter μ_b, represented by μ_1, μ_2, and μ_3. In HEBART, each terminal node b has its own overall μ_b parameter, plus one $\phi_{b,j}$ for each of the J groups in the node (not all groups need to have associated data in each terminal node). For example, terminal node 2 has an overall parameter μ_2, and intra-group parameters ($\phi_{2,1}$, $\phi_{2,2}$, $\phi_{2,3}$), one for each of the possible groups.

To define the HEBART model, let P be the number of trees, J to be the total number of groups, and Θ_p the set of node parameters at both the terminal node and sub-terminal node level. More fully, we have one set μ_p which are terminal node predictions for each tree, and one set $\phi_{p,j}$ for each group within each sub-terminal node for tree $\mathcal{T}_p$, $p = 1, \ldots, P$. Assuming we have a continuous variable of interest, the fundamental HEBART appears similar to the standard BART approach:

$$y_i = \sum_{p=1}^{P} \mathbf{G}(X_i, z_i; \mathcal{T}_p, \Theta_p) + \epsilon_i, \tag{4.3}$$

for observation $i = i, \ldots, n$ with grouping variable z_i taking values from 1 to J. In this specification, we have that $\mathbf{G}$ is the tree look-up function which allows for predictions based on covariates X_i and categorical grouping values z_i; by default $\mathbf{G}$ will return a sub-terminal node prediction value ϕ. If the grouping variable z_i

is not provided it will return a terminal node estimate μ, which we may write as $\tilde{G}$ for clarity. In other words, $\mathbf{G}$ is the function that maps each estimated tree structure to the corresponding values of the covariates X_i and finds in which terminal node of the tree each observation falls into (see, e.g, Figure 5.1, right panel, for an example). With this, the $\mathbf{G}$ function uses the tree structures to attribute the correct predicted value for each observation. Similar to an LME model, the parameters ϕ associated with the categorical grouping variable z_i provide shifts away from the overall means μ and are constrained using a normally distributed prior. As before $\mathcal{T}_p$ is the tree structure, but now Θ_p contains both the terminal node parameters (μ) and the sub-terminal node group parameters (ϕ) for tree p. As with standard BART, the noise is assumed to be distributed as $\epsilon_i \overset{\text{iid}}{\sim} N(0, \tau^{-1})$, where τ is the overall residual precision.

The μ_p and $\phi_{p,j}$ sets contain, respectively, the overall terminal-node mean parameters and the intra-group sub-terminal node mean parameters. In other words, for each tree we will have one μ_{p,L_p} set for each of their terminal nodes, and for each group within each terminal node we will have another set $\phi_{p,L_p,j}, j = 1, \ldots, J$. All parameters in Θ_p receive priors and have their corresponding posteriors, from which we sample from in the fitting algorithm. In Figure 5.1 the terminal circles represent the terminal nodes, which all have their own μ parameters. In the HEBART tree, however, we have the addition of the intra-group parameters, represented by the hexagonal symbols at the sub-terminal node level.

In standard BART a minimum node size is usually set on the number of data points that fall into each terminal node. We retain that restriction in our approach but require no such restriction for the sub-terminal node levels. As shown below, the Gibbs update for the ϕ_j terms is still available even when no residuals fall into that particular group so we can still provide group-level predictions which can be summed over trees to produce group-level predictions from every tree.

4.3.1 Prior distributions

Many of the standard BART prior distributions carry over to our situation, which desirably means that certain parametric restrictions in our model will yield an exact BART model. Specifically, we have as usual $\mu \sim N(\mu_\mu, \tau_\mu^{-1})$ for the terminal

nodes (ignoring node index subscripts for simplicity). As in standard BART, we standardize y so that $\mu_\mu = 0$ and τ_μ is simple to calibrate using the heuristics outlined above. We have $\tau \sim \text{Ga}(\nu/2, \gamma\nu/2)$ for the residual precision. We keep the tree prior and its hyper-parameters α, β to control the shape and structure of trees. For the sub-terminal node parameters, we simply set $\phi_j \sim N(\mu, \tau_\phi^{-1}/P)$ which forces each sub-terminal node to vary according to τ_ϕ around the terminal node parameter, scaled by the number of trees. Unlike τ_μ we treat τ_ϕ as a parameter to be estimated, and provide more details of this below.

For HEBART, a single sub-terminal node b in tree $\mathcal{T}_p$ has partial residuals:

$$R_{ipb} = y_i^{(b)} - \sum_{t \neq p} \mathbf{G}(X_i^{(b)}, z_i^{(b)}; \mathcal{T}_t, \Theta_t) \tag{4.4}$$

for observation i with cateogorical variable z_i. Now, let $\underset{\sim}{R}_j = \{R_{ij}, \ldots, j = 1, \ldots, J\}$ represent the full set of residuals for those observations where $z_i = j$, where we drop the dependence on terminal node and tree for simplicity of notation, and vectorize it again so we can write $R \sim N(M\phi, \tau^{-1}I)$. In this fashion, R represents all observations in that particular terminal node for that particular tree now across all groups, M is a binary matrix that allocates observations in the terminal node to groups, and ϕ represents the stacked vector of terminal node parameters for all groups in that terminal node. Marginalising first over ϕ and then over μ, we obtain a distribution on the partial residuals as $R \sim MVN(0, \Omega_R)$, where $\Omega_R = \tau^{-1}I + (P\tau_\phi)^{-1}MM^T + \tau_\mu^{-1}11^T$. This variance matrix is symmetric and can be inverted quickly when required using Woodbury and related formulae.

We give the τ_ϕ parameter, responsible for capturing the intra-group precision, a Gamma(a, b) prior. Since the value of τ_ϕ is analogous to that of a standard LME model we first fit a random intercept model to the same data using the `lme4` package (Bates et al., 2011) and write this estimated precision as $\hat{\tau}_\phi^{\text{LME}}$. We further extract the variance of this estimate via the `parameters` package (Lüdecke et al., 2020) which we write as $\hat{\sigma}^2(\hat{\tau}_\phi^{\text{LME}})$. The given mean and variance then provide simple point estimates of a and b using the standard method of moments, where we find those values such that $P(\tau_\phi < \hat{\tau}_\phi^{\text{LME}}) = 0.5$. Our approach is analogous to

the argument used in standard BART to set τ via the residual variance of a linear model fit, though there the BART residual variance is expected to fall below the linear model fit with high probability. To set our prior for $\tau \sim \text{Ga}(\nu/2, \gamma\nu/2)$, we calibrate the hyper-parameter values using the same rule as BART, but via the residual variance of the above `lme4` model fit rather than the standard linear model. In this case, our strategy aims to yield a high probability that τ is bigger than $\hat{\tau}^{\text{LME}}$, so we find prior hyperparameters such that $P(\tau > \hat{\tau}^{\text{LME}}) = 0.95$.

4.3.2 Links with standard hierarchical models

Conditional on the trees, the model can be written as a standard linear mixed effects model. We abuse the above notation slightly to write:

$$\mathbf{y}|T, \ldots \sim N(\mathbf{S_1}\boldsymbol{\phi}, \tau^{-1}\mathbf{I}) \tag{4.5}$$

where $\mathbf{y}$ is the vector of all observations and $\mathbf{S_1}$ is a binary matrix that allocates each observation to its correct tree and sub-terminal node. The number of columns of $\mathbf{S_1}$ can be large when the number of trees and/or terminal nodes is large, and changes dimension when the conditioning on the trees is removed. $\boldsymbol{\phi}$ here is the stacked vector of all sub-terminal node parameters sorted over all trees. This parameter too can be marginalized over since, within each tree, all components come from a $N(\mu, \tau_\phi^{-1}/P)$ distribution. This second marginalization yields:

$$\mathbf{y}|T, \ldots \sim N(\mathbf{S_2}\boldsymbol{\mu}, \boldsymbol{\Psi}_y) \tag{4.6}$$

where now $\boldsymbol{\mu}$ is the stacked terminal node values across all trees and $\Psi_y = \tau^{-1}\mathbf{I} + (P\tau_\phi)^{-1}\mathbf{S_1}\mathbf{S_1}^T$. $\mathbf{S_2}$ is a binary allocation matrix that allocates each observation to the terminal nodes associated with its trees. With this second marginalization there are several links with standard Bayesian mixed effects models frameworks. The model can be seen as a Gaussian Process with kernel autocorrelation function given by $S_1 S_1^T/P$, or as a standard mixed effects model with $\mathbf{y} = \mathbf{S_2}\boldsymbol{\mu} + \mathbf{S_1}\boldsymbol{\phi} + e$ with a prior on ϕ centered on zero. The standard theory of LMEs (e.g. Pinheiro and Bates, 2000) follows directly, and in the Bayesian framework, the updates for sub-terminal node parameters yields the usual partial pooling estimates exemplified by Gelman and Hill (2006); see updates for ϕ below. However, the mixing over

the trees makes any further theory considerably more complex, and we leave this as a challenge for future research.

4.3.3 Updating parameters

We use MCMC methods to update the parameters, using direct Gibbs sampling for updating the terminal node and residual precision parameters, and Metropolis updates for those for which a tractable Gibbs update is unavailable. The updates for an individual tree $\mathcal{T}$ arise by considering the distribution of the partial residuals in a single terminal node, whilst integrating out the mean and variance parameters. We use the marginalized version of R above to find an MH update for a single terminal node in a tree (on the log scale, and summed over terminal nodes b) as:

$$\log(\pi(R_1, \ldots, R_n)) \propto \sum_{b=1}^{L} \left[-\frac{1}{2} \log |\Omega_{R,b}| - \frac{1}{2} R_b^T \Omega_{R,b}^{-1} R_b \right]$$

For μ, we use the non-marginalised prior for R and the prior on μ to obtain the full conditional:

$$\mu | \ldots \sim N\left(\left(1^T \Psi_R^{-1} 1 + \tau_\mu\right)^{-1} 1^T \Psi_R^{-1} R, \left(1^T \Psi_R^{-1} 1 + \tau_\mu\right)^{-1} \right),$$

which is used to directly sampled values for the μ parameters. In this equation, we have that $\Psi_R = \tau^{-1} I + (T\tau_\phi)^{-1} M M^T$. Similarly, we also obtain a closed-form posterior distribution for ϕ:

$$\phi | \ldots \sim N\left(\left(\tau M^T M + T\tau_\phi I\right)^{-1} (\tau M^T R + T\tau_\phi \mu), \left(\tau M^T M + T\tau_\phi I\right)^{-1} \right)$$

After all trees are updated we can update τ. The τ update comes directly from the non-marginalised prior for R combined with the prior on τ:

$$\tau | \ldots \sim Ga\left(\frac{N + \nu}{2}, \frac{(y - S_1\phi)^T (y - S_1\phi) + \nu\lambda}{2} \right)$$

Last, we use a random-walk Metropolis-Hastings update for τ_ϕ, which we found to be reliable for sampling from this posterior (Brooks et al., 2011b). We use a

standard Normal for the proposal value to create τ_ϕ^* and decide to accept or reject it based on the acceptance probability $\alpha(\tau_\phi^*, \tau_\phi) = \min\left\{1, \frac{P(\mathbf{y}|\tau_\phi^*,\Theta)P(\tau_\phi^*)\Phi(\tau_\phi)}{P(\mathbf{y}|\tau_\phi,\Theta)P(\tau_\phi)\Phi(\tau_\phi^*)}\right\}$, where $\Phi(x)$ represents the CDF of the standard Normal distribution. This is used to account for the fact that we are using a proposal distribution with full support for a target with limited support.

Algorithm 2 HEBART Algorithm

Type: Metropolis within GIBBS for a hierarchical BART model

Require: y, X, grouping variable z

Ensure: Posterior distribution of trees $\mathcal{T}$, μ, ϕ, τ and τ_ϕ Initial values for α, β, σ_ϕ, τ, number of trees P, stumps $\mathcal{T}_1, \ldots, \mathcal{T}_P$, number of observations N, number of MCMC iterations $\mathcal{I}$ = burn-in+ post-burn-in, initial residual set $\mathbf{R}^{(1)} = \mathbf{y}$

for $i \leftarrow 1$ to $\mathcal{I}$ **do**

for $p \leftarrow 1$ to P **do**

1. Grow a new tree $\mathcal{T}_p^*$ tree by either growing, pruning, changing or swapping a root node
2. Calculate $\alpha(\mathcal{T}_p^*, \mathcal{T}_p) = \min\left\{1, \frac{P(R_p^{(i)}|\mathcal{T}_p^*,\tau)P(\mathcal{T}_p^*)}{P(R_p^{(i)}|\mathcal{T}_p,\tau)P(\mathcal{T}_p)}\right\}$;
3. Sample $u \sim U(0, 1]$
4. **if** $u < \alpha(\mathcal{T}_p^*, \mathcal{T}_p)$ **then do** $\mathcal{T}_p = \mathcal{T}_p^*$

for $b \leftarrow 1$ to b_p **do**: Sample $\mu_{b,p}$

for $j \leftarrow 1$ to J_{b_p} **do**: sample $\phi_{b,p,j}$

end for

end for

Update $\mathbf{R}_p^{(i)} = \mathbf{y} - \sum_{t\neq p}^{P} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_i}, \mathcal{T}_t, \Theta_t)$

Update $\hat{f}_{ij} = \sum_{p=1}^{P} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_i}, \mathcal{T}_p, \Theta_p)$

end for

Sample τ

Sample τ_ϕ:

Sample a value $d \sim N(0, \sigma^2_{\tau_\phi})$ and make $\tau_\phi^* = \tau_\phi + d$

Calculate $\alpha(\tau_\phi^*, \tau_\phi) = \min\left\{1, \frac{P(\mathbf{y}|\tau_\phi^*,\Theta)P(\tau_\phi^*)\Phi(\tau_\phi)}{P(\mathbf{y}|\tau_\phi,\Theta)P(\tau_\phi)\Phi(\tau_\phi^*)}\right\}$;

Sample $u \sim U(0, 1]$

if $u < \alpha(\tau_\phi^*, \tau_\phi)$ **then do** $\tau_\phi = \tau_\phi^*$

end for

4.4 Applications & Results

This section describes the fitting of HEBART to some example data sets as we compare across different models, including: a Linear Mixed-Effects (LME) model, fitted using the `lme4` (Bates et al., 2011) package; in `R` (R Core Team, 2018) and standard BART, fitted using the `dbarts` (Dorie, 2020) `R` package, with the default values for the number of trees (200), number of posterior samples (1000), number of burn-in samples (100). The main performance results presented are calculated on out-of-sample values for a certain number of test sets. Since BART cannot incorporate mixed effects, other than as part of the main covariate set, we partition the results into BART including grouping information, which we expect to perform well though it does not solve the hierarchical modeling problem, versus BART without grouping information, which usually performs worse than our approach since it lacks all the data.

The package created to run our HEBART algorithm is available in `R` (R Core Team, 2018). All functions are available at `https://github.com/brunaw/hebartBase`, where the main package is stored, and `https://github.com/brunaw/hebart-experiments` contains the code for replicating all the examples shown here, with their corresponding auxiliary files.

4.4.1 Simulation experiments

We first experiment with HEBART on a simple simulation scenario, where the response variable y is simulated as a sum of a tree structure and an intra-group parameter, which here we call $\phi_j, j = 1, \ldots, 5$ and are simulated from a $N(0, 5^2)$ to create highly variant values. By splitting the simulated covariates $X_1, X_2,$ and X_3 at random points, we sample $n = 750$ values as

$$\begin{aligned} y_{ij} \sim & N\left((a\,\mathbf{I}(X_1 < 1)\mathbf{I}(X_2 < 0.2) + b\,\mathbf{I}(X_1 < 1)\mathbf{I}(X_2 >= 0.2) + \right. \\ & \left. c\,\mathbf{I}(X_1 >= 1) + d\,\mathbf{I}(X_3 >= 2)\right) + \phi_j, \tau^{-1} = 1, \end{aligned} \tag{4.7}$$

where we have $n_j = 150$ for all 5 groups and (a, b, c, d) are randomly sampled from a $N(0, 3^2)$. This experiment uses 10 different train and test sets as our cross-validation setting (Refaeilzadeh et al., 2009).

To these simulated datasets, we apply three different algorithms (LME, Standard BART and HEBART) to the 10 training sets and make predictions for the 10 testing sets. In Figure 4.2, we can see that HEBART produces the smallest root mean squared errors for both the train and test sets, as its averages are the closest to zero, followed by LME and standard BART. We also observe that HEBART has the least varying RMSE values on the test set, implying there is less uncertainty around the predictions for new data.

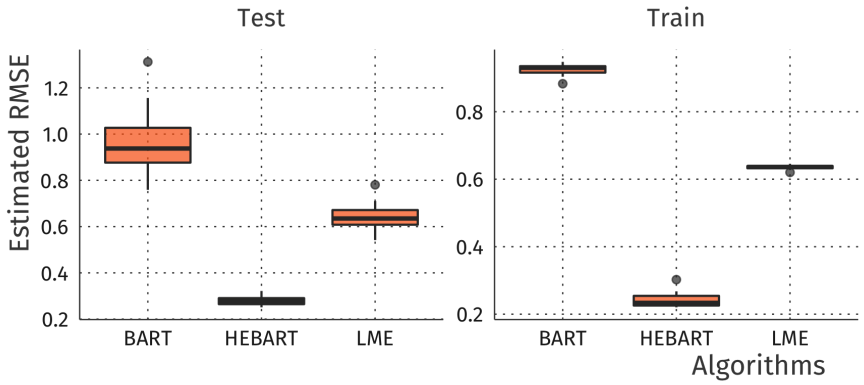


Figure 4.2: Boxplots of RMSE for testing and training sets for the 3 algorithms fitted on simulated data based on Equation (7). HEBART is the best-performing algorithm in either stage.

In a second simulation setting we create y with a direct HEBART structure, by first sampling a μ value for each terminal node, along with one ϕ value for each group within the created terminal nodes, where $\tau_\mu = 3, \tau_\phi = 3$ and $\tau = 1$. The tree structure uses the sum of two simple trees, which are both based on two covariates $(X_1, X_2) \sim \text{Normal}(0, 1)$, and uses the previously sampled parameters to create the simulated response y. By splitting the simulated covariates X_1, X_2 at random points, we sample $n = 1000$ values as

$$y = G(X_1, Z_1, T_1, \Theta_1) + G(X_2, Z_2, T_2, \Theta_2) + \epsilon \tag{4.8}$$

where $\epsilon \sim N(0, 1)$. Tree T_1 consists of three terminal nodes with the partitions

$I(X_1 < 0, X_2 < 0)$, $I(X_1 < 0, X_2 >= 0)$, $I(X_1 >= 0, X_2 >= 0)$ respectively, having terminal node values μ_{b_1} sampled from a $N(0, 3^{-1})$, and group terminal node values sampled from a $N(\mu_{b_1}, (2/3)^{-1})$, $b_1 = 1, \ldots, 3$, as we have 3 terminal nodes. For tree T_2 the partitions are only $I(X_2 < 0.5)$ and $I(X_2 >= 0.5)$, having terminal node values μ_{b_2} sampled from a $N(0, 3^{-1})$, and and group terminal node values sampled from a $N(\mu_{b_2}, (2/3)^{-1})$, $b_2 = 1, \ldots, 2$.

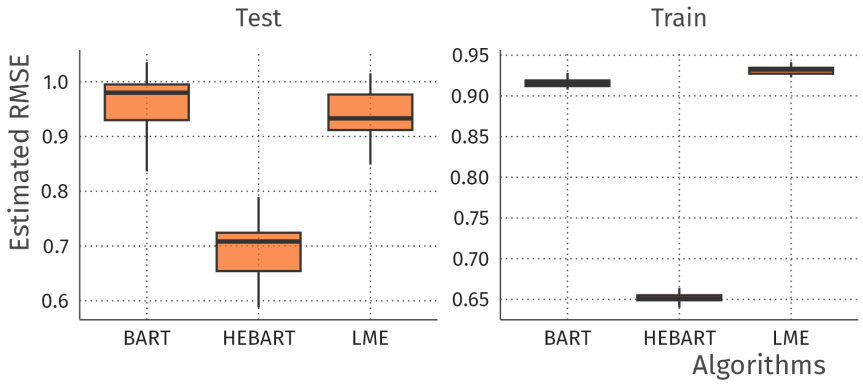
The results of this simulation are shown in Figure 4.3. Once again, we observe HEBART to be the best-performing algorithm, as its test RMSE average is lower than its competitors. We also include here the prior and posterior densities of τ_ϕ one of the runs of the HEBART model. The two densities are markedly different, with the posterior having a distinctly larger variance (lower precision) associated with the grouping information. This is not necessarily surprising given that the grouping information for this simulation scenario is specified at the sub-terminal node level rather than through an additive effect as is used in an LME.

4.4.2 Real data sets

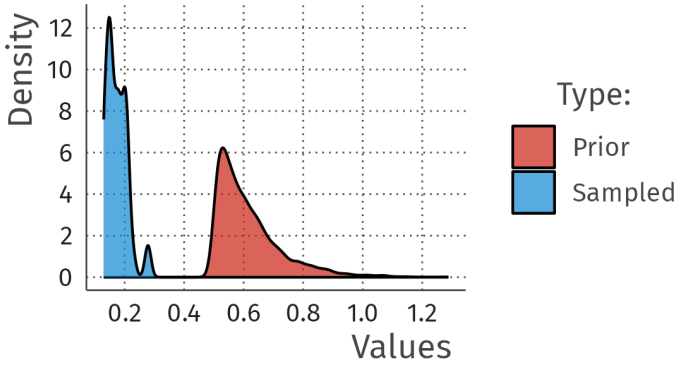
4.4.2.1 Sleep study data

This dataset consists of the first 10 days of a sleep study (Belenky et al., 2003). The response variable is the average reaction time per day (in milliseconds), and the covariate is the number of days of sleep deprivation, for 18 subjects. Linear mixed-effect models (Pinheiro and Bates, 2000) are often applied to this data (Bates et al., 2011), so we can directly compare our results.

We create a 20-fold cross-validation and evaluate the model on the left-out data. Our HEBART model is set to use 10 trees, has hyperparameters $\alpha = 0.95$ and $\beta = 2$, and all the other prior hyperparameters are set as described above. For the MCMC sampling, we use 1500 iterations with 250 iterations of burn-in. We also fit an LME (Pinheiro and Bates, 2000) and a standard BART model (Chipman et al., 2010b) to compare our results. Our main interest is in the performance of the predictions and the estimate of the group-level random effect standard error, though this is not available in the BART model since it has no such parameter. However, as described above the grouping variable is sub-optimally included as



(a) Boxplots of RMSE for testing and training sets for the 3 algorithms fitted on simulated data



(b) Prior and posterior densities for τ_ϕ for one run of HEBART

Figure 4.3: (a) For RMSE, HEBART exhibits superior performance in the test and train sets compared to its two competitors for the second simulation setting; (b) The prior and posterior densities show how the two distributions significantly differ in this case. It appears that the information in the data pushes the posterior of τ_ϕ to be lower than indicated by the prior distribution obtained from a random intercepts model fit.

one of the covariates.

In Figure 4.4 we show the HEBART, BART and LME predictions for each group ID, for predictions created when those observations were not included in the train-

ing set. In other words, we are seeing the average predictions for the test sets, split by group ID. The plot also shows the true observations as green dots for comparison. We can see for almost all IDs, the HEBART predictions are the ones closest to the true values, especially for more difficult cases, such as IDs 335, 332, 351 and 351. Overall, HEBART is able to adapt better to changes in each of the individual group patterns. In the same Figure, we also have the RMSE table for each model, with their corresponding empirical 95% confidence intervals, calculated using the results from the 20 train and test sets. From this table it is clear that HEBART produces the best results, as both the train and test RMSEs are the lowest of the three methods, with similarly low values for the confidence intervals.

4.4.2.2 Gapminder data

Another standard dataset used to exemplify mixed effects modeling is the Gapminder data (Lang, 2011). For our experiment, the subset of the data consists of the life expectancy values (in years) for 20 different countries, from 1950 to 2018. We use life expectancy as the response and country as the grouping variable with year as the sole covariate. We separate the dataset into 10 different training and test sets, where the testing set is composed of all the observations for 15 sampled years (for all countries), and the corresponding training set is composed of the observations for the remaining data. The idea behind this is, for each resample, to fully remove a set of years from the training set to make it harder for the model to predict for such years, since it has no information about what happened in the removed years. We also compare our model against LME, BART and BART using the country as a covariate. We expect the latter to perform well since it has all the information, but we remind the reader that this method does not have the added advantage of HEBART in being able to predict with fully or partially missing grouping values, as HEBART reduces to BART when that happens, which is a fairly competitive algorithm in most scenarios.

Figure 5.4 shows the results for this experiment as an RMSE table and the depiction of the predicted values in comparison to the true observations. Note that we have 10 training and test sets, and what we show in the Figure are the average predictions for the 10 test sets. So, when we bind all the predictions for all the 10

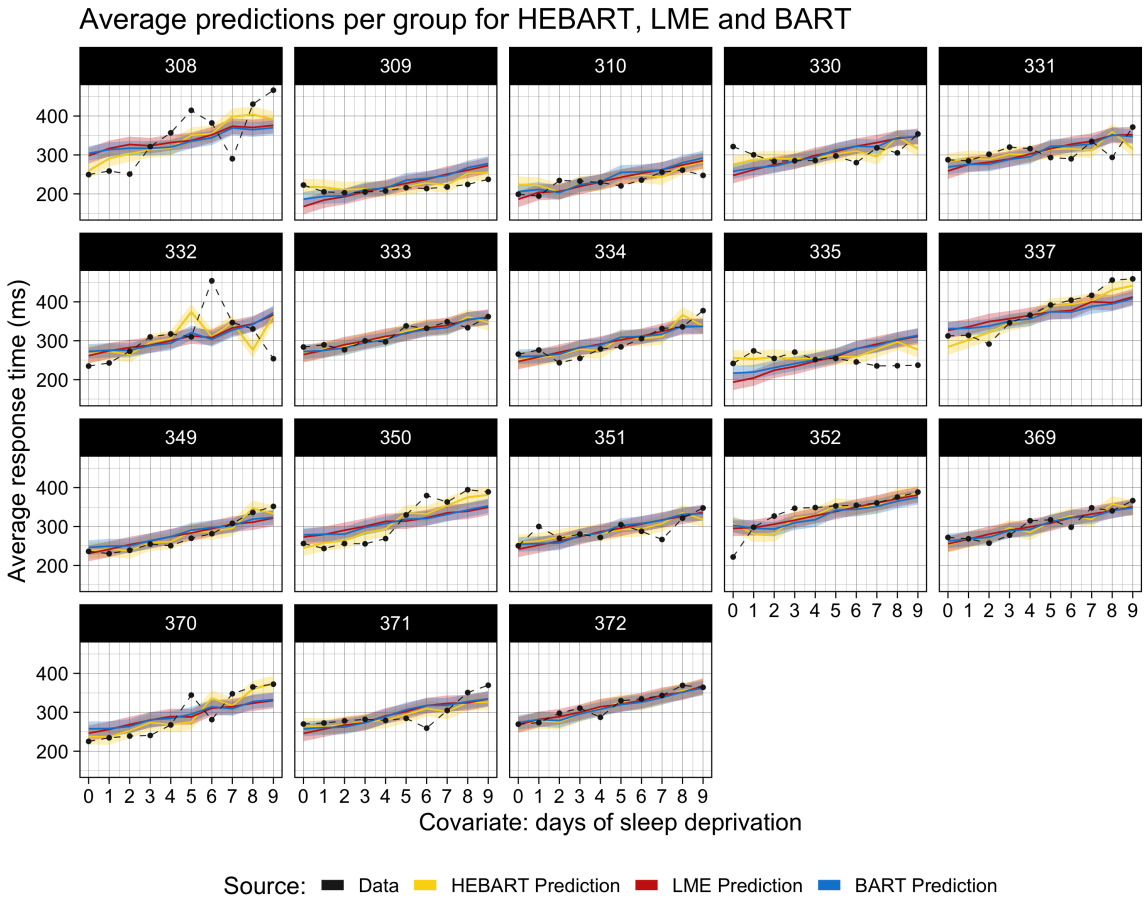
test sets, we have at least one prediction for each year of all countries. As above, the HEBART prediction is very close to the actual values due to the increase in flexibility of the predictions from the non-parametric structure. The performance of the BART model that uses the country as a covariate (BART+Group) is slightly superior but as stated above, this is not unexpected since the BART+Group misses the fundamental advantage of HEBART in separating out (and potentially removing) the country-level variability. Our model, on the other hand has the advantage of being able to predict when grouping variable information is missing, but still retains the prediction performance. Lastly, in Figure 4.6 we see the convergence plots for τ, just to show that the chains are stable, even though they do not agree for all runs of this experiment.

4.5 Conclusions

We have provided a new extension of Bayesian Additive Regression Trees that allows for structured data to be used appropriately at the model fitting and prediction stage. Where a grouping variable is present, we are able to provide predictions at both the group level and the level above it. The flexible tree structure thus allows us to produce excellent predictions without the need to specify how the grouping variable is included in the model structure. However, we still retain the ability to report and remove the group-level variability by having a parameter that represents the group-level standard deviation.

In simulation-based and real data studies we have shown that the model performs better than other common linear and mixed effects approaches or BART itself. Our approach is not comparable to many other standard Bayesian machine learning methods due to their inability to handle the grouping information other than as a naïve covariate. In the real data examples of Section 4.4.2.1 and 4.4.2.2 we have demonstrated that HEBART performs well against LME modelling strategies despite these being the archetypal examples of datasets where linear mixed-effects models would fit well. Finally, we see many potential extensions of our approach, including: (1) extending the hierarchical data structure to multiple or nested grouping variables; (2) explicitly modeling joint random effects using the multivariate normal distribution and so estimating covariances between grouping

variables; (3) including recent BART extensions, such as SOFT-BART (Linero, 2018a) and MOTR-BART (Prado et al., 2021a) which can substantially enhance the predictive capabilities of the model;

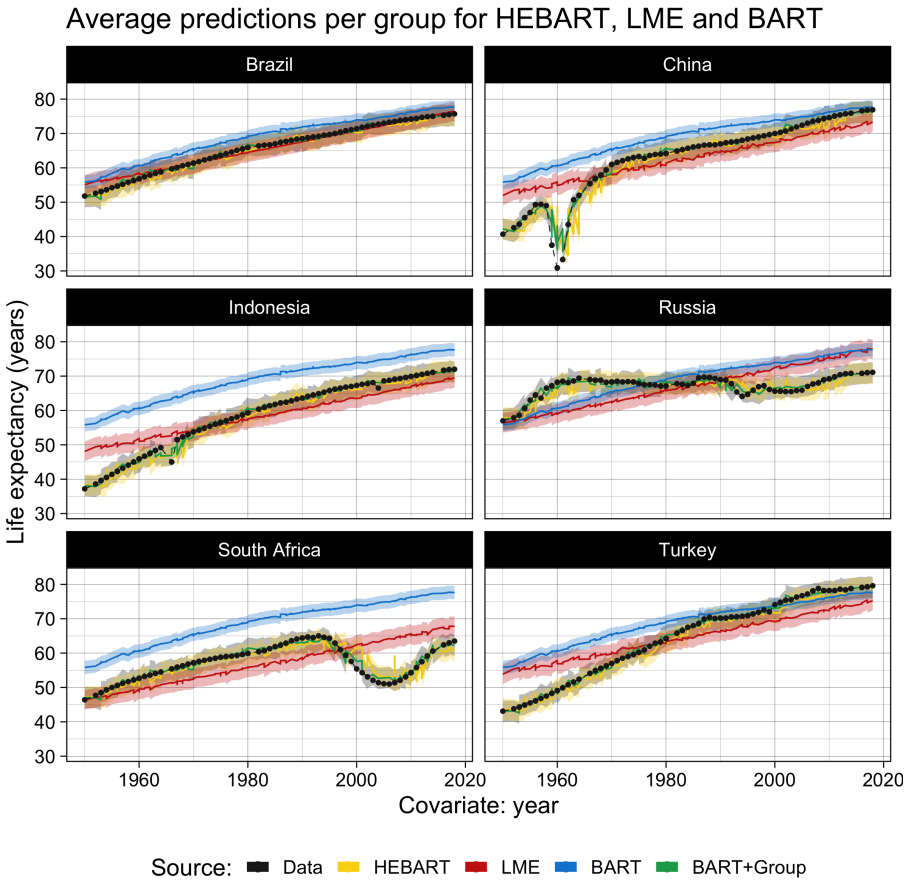


(a) Average predictions for the groups in the sleep study dataset

Source	HEBART	LME	BART+Group
Test set	27.7 [8.22, 47.2]	32.4 [15.8,49]	32.3 [13.3, 51.2]
Train set	16.9 [10.1,23.7]	29.3 [28.3,30.3]	28.1 [26.6,29.6]

(b) Average train and test set RMSE values, with the corresponding empirical 95% confidence intervals

Figure 4.4: (a) Average predictions for each patient ID in the sleep study data, using HEBART, LME and BART with confidence/credible intervals. HEBART produces predictions closer to the actual true data, including the most challenging IDSs; (b) Average test RMSE values, with empirical 95% confidence intervals. The lowest values are for HEBART.



(a) Average out of sample predictions for selected countries in the gapminder dataset

Source	HEBART	LME	BART	BART+Group
Test set	1.33 [0.77, 1.88]	3.59 [2.84, 4.33]	7.53 [6.53, 8.52]	0.815 [0.43, 1.20]
Train set	0.793 [0.162, 1.42]	3.57 [3.38 , 3.77]	7.47 [7.20, 7.73]	0.351 [0.310, 0.391]

(b) Average training and test RMSE values, with the corresponding empirical 95% confidence intervals

Figure 4.5: (a) Average predictions (considering the 10 test sets) for 6 selected countries in the gapminder dataset, using HEBART, LME, BART, and BART using group as a covariate, with confidence/credible intervals. HEBART and BART+Group both produce predictions closer to the actual true data for all cases shown; (b) Average test RMSE values, with empirical 95% confidence intervals for the mean RMSE. The table confirms the lowest values for HEBART, with neither the training value not even intersecting with the other ones, except for HEBART and LME.

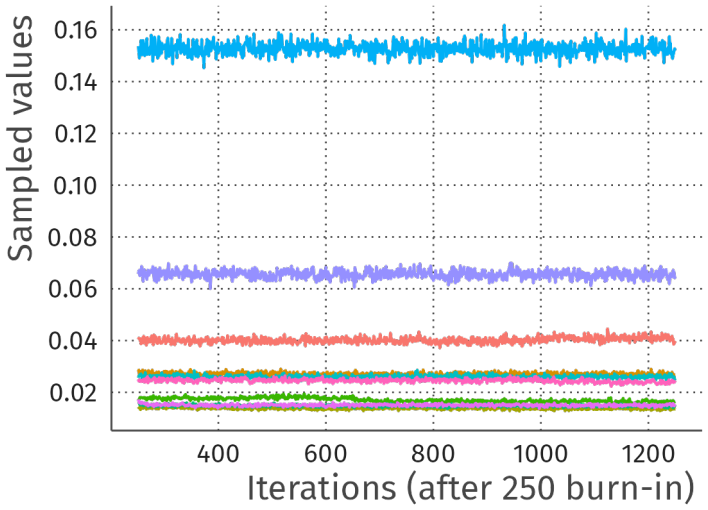


Figure 4.6: Convergence plots of posterior distributions of τ for the 10 runs; all chains seem to be stable.

CHAPTER 5

Hierarchical Embedded Bayesian Additive Regression Trees for Crossed and Nested Random Effects

Bayesian Additive Regression Trees (BART; Chipman et al., 2010b) is a now standard probabilistic machine learning approach based on sums of regression trees. It has been widely used and extended (e.g. for categorical, count, and multinomial regression (Murray, 2021; Kindo et al., 2016b) and quantile regression (Kindo et al., 2016a)). Recently, it has also been extended to produce Hierarchical Embedded BART (HEBART) which allows for hierarchical data structures (Bates et al., 2011, or 'random effects' in mixed model parlance;) to be used in BART, as previous shown in this document. Whilst previous approaches to incorporating hierarchical data in BART have been proposed (even in the original 2010 paper) these have involved simply adding an extra term onto the additive model. By contrast, HEBART places the random effects inside the terminal node structures (hence the term embedded) and so provides a far richer modelling structure that removes the need for difficult model comparison questions about where to place the random effects in standard regression models, e.g. in intercepts or slopes.

In general, HEBART allows for greater flexibility in the fitting of the regression trees as it allows for predictions at multiple layers of a data hierarchy. We start by considering the maximally simple situation of triplets (y, X, z) where y is a response, X is a set of covariate values, and z is a categorical grouping variable with levels 1 to J. In traditional BART, predictions are only available at the terminal node level (i.e. using only y and X), and so each observation with the same covariate values must be given the same predicted value. In HEBART the single grouping variable z allows for predictions to be created at both the terminal node level (depending on only X) and including that of the categorical grouping variable beneath it (depending on X and z). See Section 5.1 for a complete mathematical description of the approach. Thus if the covariate values are available but not the grouping variables, the model can still produce predictions with quantified uncertainties. If both covariates and grouping variable are available then the predictions use all available information.

In this paper we extend HEBART in two ways. These are:

1. Crossed Random Effects HEBART (CHEBART) which allows for multiple grouping variables in the same model, i.e. $z_1, z_2, \ldots$.
2. Nested Random Effects HEBART (NHEBART) which accounts for multiple nested grouping variables, i.e. z_{ij} where for each level i there are J_i sub-levels.

For (1) we allow a proportion of the HEBART trees to be used with each grouping variable, thus allowing them to have their effect captured in the model, each with an associated variance parameter. For the second model, we also have the presence of more than one grouping variable, but now in a nested rather than a crossed fashion. The trees used in this second model have further sub-terminal nodes representing the sub-grouping of the categorical factors. We describe both of these models and outline fitting algorithms and their performance on simulated and real-world data sets. A schematic of the differences between the approaches, more fully described in Section 5.3, is shown in Figure 5.1.

Our paper is structured as follows. In Section 5.1 we introduce the BART and HEBART mathematically and discuss some extensions that have already been

proposed. In Section 3 we outline our two HEBART extensions and explain how they generalize HEBART and comprise an important addition to the overall BART family of models. In Section 4 we demonstrate the performance of the two methods on some simulated and real-world data. Finally, Section 5 discusses some of the potential future research areas and drawbacks of our approach.

5.1 An introduction to BART and HEBART

5.1.1 The BART model

The Bayesian Additive Regression Tree (Chipman et al., 2010b) is a fairly recent algorithm, which assumes that the generating system of a continuous random variable $\mathbf{y} = [y_1, \ldots, y_n]$ can be approximated by a sum of regression trees. In a standard regression setting the BART model is usually written as:

$$y_i = \sum_{p=1}^{P} \mathbf{G}(X_i; \mathcal{T}_p, \Theta_p) + \epsilon_i, \ \epsilon_i \sim N(0, \tau^{-1}), \tag{5.1}$$

for observations $i = i, \ldots, n$, where P is the fixed number of trees, X_i represents the set of covariates; $\mathcal{T}_p$ is a tree structure, and Θ_p represents a set of terminal node parameters. The $\mathbf{G}$ function maps each tree structure to the covariates space, returning a terminal node prediction from Θ by passing the values of X_i through the tree T_p and allocating each observation to each tree node. Θ_p consists of a set of $\mu_{p,l}$ parameters for each of the $l = 1, \ldots, L_p$ terminal nodes in tree p. These tree-level predictions are summed together across all the trees to give an overall predicted value. The residual term ϵ is assumed normal with residual precision τ. We write the set of all trees and parameters as $\mathcal{T}$ and Θ, respectively. The joint posterior distribution of the trees and all the parameters is then given by:

$$P(\mathcal{T}, \Theta, \tau | \mathbf{X}, \mathbf{y}) \propto \left[\prod_{p=1}^{P} \prod_{b=1}^{b_p} \prod_{i:\mathbf{x}_i \in \mathcal{D}_{p,b}} p(y_i | \mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau) \right] \times \left[\prod_{p=1}^{P} \prod_{b=1}^{b_p} p(\mu_{p,b} | \mathcal{T}_p) p(\mathcal{T}_p) \right] p(\tau), \tag{5.2}$$

where $p(y_i|\mathbf{x}_i, \mathcal{T}_p, \Theta_p, \tau)$ is the normally distributed likelihood as defined in Equation 5.1, and $\mathcal{D}$ represent the regions in the covariates space (e.g. tree nodes). The term $p(\mu_{p,b}|\mathcal{T}_p)$ is the prior distribution on the terminal node parameters across each terminal node b in each tree p. In addition, $p(\mathcal{T}_p)$ is the prior distribution on the tree structure, and $p(\tau)$ is the prior on the residual precision. The prior distribution on the trees proposed by Chipman et al. (2010b) is created by applying a separate term for each node in the tree and considering both the probability of a split as well as the probability of a new splitting variable being chosen. For an entire tree $\mathcal{T}_p$, we have:

$$P(\mathcal{T}_p) \propto \prod_{\eta \in L_{\mathcal{T}}} (1 - P_{SPLIT}(\eta)) \prod_{\eta \in L_{\mathcal{I}}} P_{SPLIT}(\eta) \prod_{\eta \in L_{\mathcal{I}}} P_{RULE}(\rho|\eta).$$

where $L_{\mathcal{T}}$ and $L_{\mathcal{I}}$ represent the sets of terminal and internal nodes, and ρ is a generic splitting value. The probability of a node being non-terminal is given by $P_{SPLIT} = \alpha(1-d_\eta)^{-\beta}$, where d_η denotes the depth of the node η. The recommended values for the hyperparameters are $\alpha \in (0, 1)$ and $\beta > 0$, which control the depth and bushiness of the trees. For the probability of the new splits, $P_{RULE}(\rho|\eta, \mathcal{T}) = \frac{1}{p_{adj}(\eta)} \frac{1}{n_{j.adj}(\eta)}$ where $p_{adj}(\eta)$ represents how many predictors are still available to split on in node η, and $n_{j.adj}(\eta)$ how many values in a given predictor are still available.

The prior distribution for the terminal node and overall parameters in the standard regression case is denoted by $p(\Theta, \mathcal{T})$, and it has a standard conjugate form:

$$\mu_1, \ldots, \mu_b|\tau_\mu, \mu_\mu, \mathcal{T} \sim \mathcal{N}(\mu_\mu, \tau_\mu^{-1}),$$

where τ_μ^{-1} and μ_μ are chosen such that a high density of this distribution is apportioned to the range $[y_{min}, y_{max}]$ interval, by setting $P\mu_\mu - k\sqrt{P(\tau_\mu^{-1})} = y_{min}$ and $P\mu_\mu + k\sqrt{P(\tau_\mu^{-1})} = y_{max}$, for some value of k. The response y is usually standardized before the model is run which allows for reasonable guesses as to the hyper-parameter values of μ_μ and τ_μ, though these too can be estimable parameters. Finally, the residual precision prior is set as $\tau \sim \text{Gamma}(\nu/2, \gamma\nu/2)$, where

γ and ν are fixed. An oft-used tactic is to set the two hyper-parameters such that the BART residual precision is greater than an equivalent precision value from a standard linear regression model applied to the same data with a high probability. We follow the same guidance in our extension to the model as outlined below.

Since its creation, BART has been applied to a wide variety of different areas: credit risk modeling (Zhang and Härdle, 2010), survival data analysis (Sparapani et al., 2016, 2020), ecology and evolution modelling (Carlson, 2020), weather and avalanche forecasting (Blattenberger and Fowles, 2014), and genetics (Waldmann, 2016). A popular approach is its use in causal inference (Hill, 2011; Hahn et al., 2020), where BART produces accurate estimates of average treatment effects and is competitive even with the true data generating model. Many fundamental extensions to the standard BART model have been proposed, including BART for categorical, count, and multinomial regression (Murray, 2021; Kindo et al., 2016b) and quantile regression (Kindo et al., 2016a). This was followed by the proposal of BART that adapts to smoothness and sparsity (Linero and Yang, 2018b), models for high-dimensional data and variable selection (Linero, 2018a), BART for zero-inflated and semi-continuous responses (Linero et al., 2020b) and an extension proposed by (Hernández et al., 2018), where the authors combine BART with Bayesian Model Averaging to obtain posterior distribution more efficiently when there is a large number of variables available. Quite a few more extensions have also been proposed, showing how BART is becoming popular and useful in many different settings, such as heterocedastic data (Pratola et al., 2020a), the estimation of monotone and smooth surfaces, (Starling et al., 2020), varying coefficient models (Deshpande et al., 2020), semiparametric BART (Prado et al., 2021b), and a combination of BART with model-trees (Prado et al., 2021a). Some of the mathematical properties of BART, including a deep review of the BART methodology can be found in Linero (2017), and some more general theoretical results in Ročková and van der Pas (2020); Ročková and Saha (2019).

5.1.2 Hierarchical Embedded Bayesian Additive Regression Trees (HEBART)

The HEBART approach merges the ideas from traditional Bayesian hierarchical modeling (Gelman and Hill, 2006) and linear mixed effects models (Pinheiro and Bates, 2000) with BART, as the model allows each tree to have an extra split on the terminal nodes, which corresponds to a random intercept for each member of a categorical predictor variable z, taking values from 1 to J groups. Sub-terminal node level parameters are introduced and written as $\phi_{b,j}$, as the estimate for group j in terminal node b. In relation to standard BART, the added parameters allow for a group-specific prediction as well as an overall terminal node prediction μ. This flexibility means that the user is not required to specify where the random effect is included, for example, as an intercept or a regression slope. We can thus fit Bayesian Additive Regression Trees to any grouped data where there is a categorical predictor, such as longitudinal data, repeated measures data, multilevel data, and block designs. Figure 5.1 (panel a) shows a standard HEBART tree, where we have both the terminal node and sub-terminal-node parameter levels.

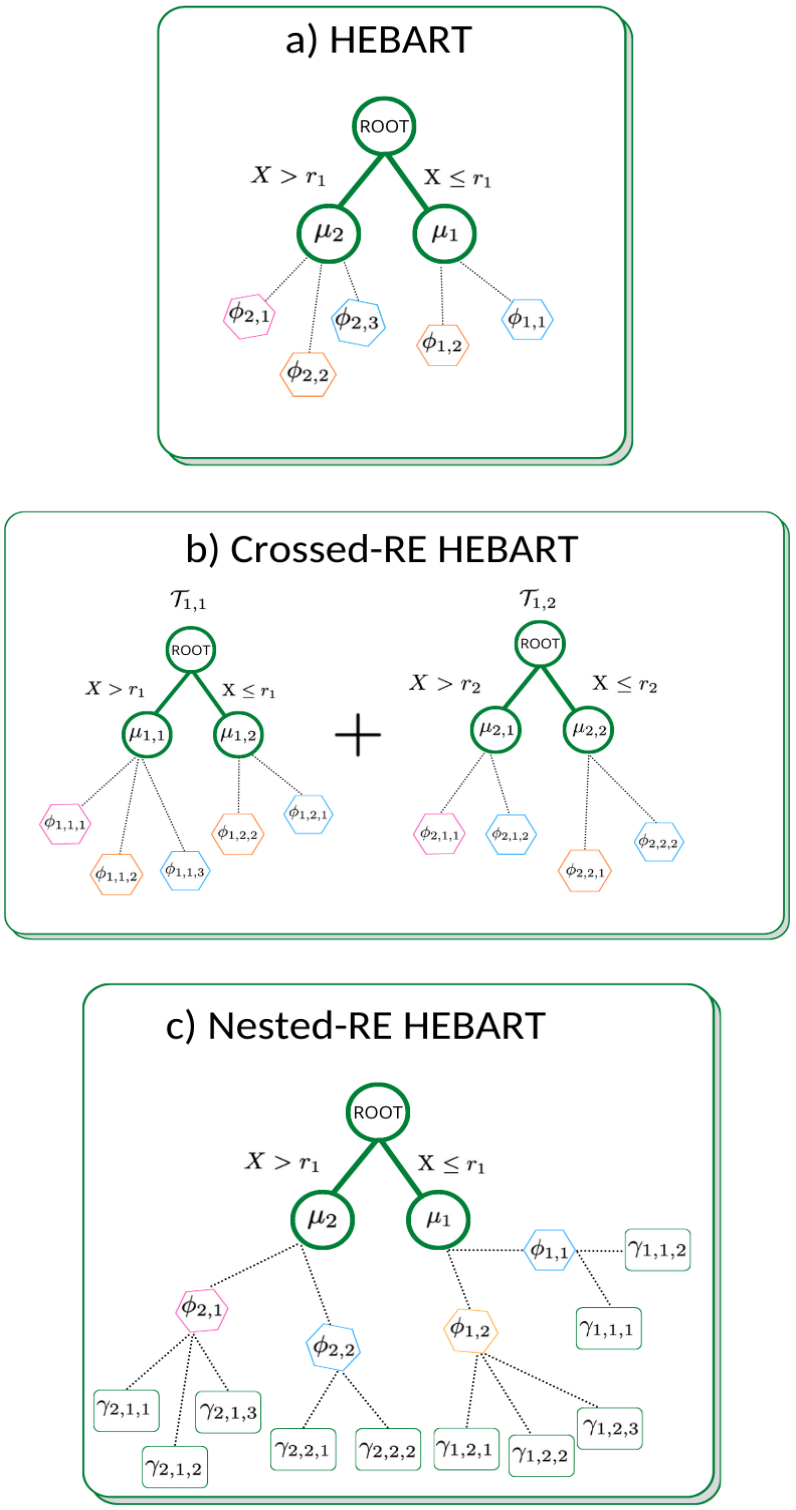


Figure 5.1: Top panel: a HEBART tree using one covariate X and a grouping variable z with three levels. In HEBART, each terminal node b has its own overall μ_b parameter, plus one $\phi_{b,j}$ for each of the J groups in the node (not all groups need to have associated data in each terminal node); Middle panel: a Crossed-RE HEBART tree, where we have an extra index for the μ and ϕ, used to represent the grouping variable index; The $\mathcal{T}$ are indexed accordingly to the tree and group index; Bottom panel: a Nested-RE HEBART tree, which has yet another node sub-level, to represent the nested groups structure (we have sub-nodes that have different numbers of sub-sub-nodes).

To define the HEBART model, let P be the number of trees, J to be the total number of groups, and Θ_p the set of node parameters at both the terminal node and sub-terminal node level. We have one set μ_p of size b_p (the number of terminal nodes in tree p), which are terminal node predictions for each tree, and one set $\phi_{p,j}$ for each group within each sub-terminal node for tree $\mathcal{T}_p$, $p = 1, \ldots, P$. In the regression case, the fundamental HEBART function can be written as:

$$y_i = \sum_{p=1}^{P} \mathbf{G}(X_i, z_i; \mathcal{T}_p, \Theta_p) + \epsilon_i, \tag{5.3}$$

for observation $i = i, \ldots, n_j$. The values of the grouping variable z_i take values $j = 1, \ldots, J$ according to the group of observation i. In this specification, we have that $\mathbf{G}$ is the tree look-up function which allows for predictions based on covariates X_i. The categorical grouping variable z_i provides information for the sub-terminal node values only, and is not included in the tree splits, similar to if it were included in the intercept of a standard random effect in a linear mixed model. The key difference to standard BART is that Θ_p contains both the terminal node parameters and the sub-terminal node group parameters for tree p, and the tree lookup function G can provide both the terminal node and sub-terminal node predictions as required. Similar to BART, the noise is assumed to be $\epsilon_i \overset{\text{iid}}{\sim} N(0, \tau^{-1})$, where τ is the residual precision.

The prediction parameter set $\Theta_p = \{\mu_p, \phi_{p,j}\}$ contains, respectively, the overall terminal-node mean parameters and the group-level sub-terminal node parameters. Thus each tree will contain a set of μ_p parameters of size b_p where b_p is the number of terminal nodes in tree p, and a set of $J \times b_p$ parameters $\phi_{p,j}$ for each group in each sub-terminal node. All parameters in Θ_p are given prior distributions which can be obtained in closed form using standard Gibbs sampling.

For the terminal nodes HEBART uses the standard $\mu \sim N(\mu_\mu, \tau_\mu^{-1})$ and $\tau \sim \text{Ga}(\nu/2, \gamma\nu/2)$ for the residual precision. For the sub-terminal node parameters $\phi_j \sim N(\mu, \tau_\phi^{-1}/P)$ which forces each sub-terminal node to vary according to τ_ϕ around the terminal node parameter, scaled by the number of trees. The tree

prior and its hyper-parameters α, β are the same as in BART. The remaining key prior distribution is that of τ_ϕ which represents the intra-group precision at the terminal node level. In HEBART this is given a Gamma(a, b) prior. Analogous to the setting of the τ parameter in BART, the values a and b chosen by first fitting a simple random intercept model on the observed data to provide an estimate of $\hat{\tau}_\phi^{\text{LME}}$ and $Var(\hat{\tau}_\phi^{\text{LME}})$, and subsequently setting a and b so that $P(\tau_\phi < \hat{\tau}_\phi^{\text{LME}}) = 0.5$.

5.1.3 Fitting the HEBART model

Tree updates in both BART and HEBART are based on the calculation of partial residuals. For the full details of how to fit the HEBART model see Wundervald et al. (2022); however we provide a brief summary here. For HEBART in a single terminal node b in tree p the partial residuals are:

$$R_{ibp} = y_i^{(b)} - \sum_{t \neq p} \mathbf{G}(X_i^{(b)}, z_i^{(b)}; \mathcal{T}_t, \Theta_t) \tag{5.4}$$

for observation i. We drop the dependence on terminal node and tree for brevity, writing the generic set of partial residual in a terminal node as simply R, or as R_j for those just in group j. We treat this set as a vector so we can write $R \sim N(M\phi, \tau^{-1}I)$. M here is a binary matrix that allocates each observation in the terminal node to its associated group, and ϕ represents the stacked vector of terminal node parameters for each groups in that terminal node. Marginalising first over ϕ and then over μ we obtain a distribution on the partial residuals as $R \sim MVN(0, \Omega_R)$ where $\Omega_R = \tau^{-1}I + (P\tau_\phi)^{-1}MM^T + \tau_\mu^{-1}11^T$. This variance matrix is symmetric and can be inverted quickly when required using Woodbury and related formulae.

Since the above can be computed for each terminal node, the total marginal log likelihood for each tree can be created as a sum of the $MVN(0, \Omega_R)$ pdfs. Trees can then be updated using the standard BART moves: grow, prune, change, and swap (Chipman et al., 2010b). Once a tree is accepted the terminal node and precision parameters can be updated using their closed-form full conditionals:

$$\mu|\ldots \sim N\left(\left(1^T\Psi_R^{-1}1+\tau_\mu\right)^{-1}1^T\Psi_R^{-1}R, \left(1^T\Psi_R^{-1}1+\tau_\mu\right)^{-1}\right),$$
$$\phi|\ldots \sim N\left(\left(\tau M^TM+P\tau_\phi I\right)^{-1}(\tau M^TR+P\tau_\phi\mu), \left(\tau M^TM+P\tau_\phi I\right)^{-1}\right),$$

where $\Psi_R=\tau^{-1}I+(P\tau_\phi)^{-1}MM^T$.

The precision parameter τ_ϕ has to be updated using Metropolis-Hastings using a random walk proposal. The residual precision parameter can be updated using Gibbs:

$$\tau|\ldots \sim Ga\left(\frac{N+\nu}{2}, \frac{(y-\hat{y})^T(y-\hat{y})+\nu\lambda}{2}\right)$$

where $\hat{y}$ is the vector of predicted values at the sub-terminal node level. For given trees, the model can be seen as a mixture of linear mixed effects models (see Section 3.2 in Wundervald et al. (2022)).

5.2 Extending the HEBART model

This section proposes and discusses two different extensions to the HEBART model. Crossed random effects HEBART allows for the existence of more than one non-linked grouping variable. Nested random effects HEBART allows for multiple grouping variables where one grouping variable is nested (and unique) inside another.

5.2.1 Crossed Random Effects HEBART (CHEBART)

In this model each random effect is allocated to be used in a proportional number of trees, meaning that $\frac{P}{K}$ trees are fit for each of the K grouping variables. Many of the definitions from HEBART persist in CHEBART except that now we have:

- k as the index of grouping the variables,
- K as the total number of grouping variables,
- J_k as the number of levels for grouping variable k

- $\mathcal{T}_{p,k}$ as the p-th tree structure for the grouping variable k,
- P_k as the number of trees for grouping variable k, initially taken as $\frac{P}{K}$,
- z_{ki} as the value of grouping variable k for observation i.
- $\phi_{t_k bk}$ is the sub-terminal node value for tree t_k (associated with a specific grouping variable k), terminal node b, and group k.

For a single terminal node we now have a set μ_{p_k} which are the terminal node predictions for each tree p_k, and an associated vector ϕ_{p_k} for each group within each sub-terminal node for tree $\mathcal{T}_{p_k}$, $p = 1, \ldots, P_k$. Figure 5.1 (panel b) provides a schematic for how this model works in that we have an extra index for each of the parameters in comparison to standard HEBART. With these definitions, assuming we have a continuous variable of interest, we can write the main CHEBART equation as:

$$y_i = \sum_{k=1}^{K} \sum_{p_k=1}^{P_k} \mathbf{G}(X_i, z_{ki}; \mathcal{T}_{p_k}, \Theta_{p_k}) + \epsilon_i, \tag{5.5}$$

for observation $i = i, \ldots, n$. G here provides the terminal node and sub-terminal node predictions for each given tree, exactly as in HEBART. The only change is that a portion of the trees use differing grouping variables.

The fitting algorithm for CHEBART differs slightly from that of HEBART. First, instead of fitting the P trees using a single grouping variable, we create $*\frac{P}{K}$ for each grouping variable. We sample both μ and ϕ from the posterior distribution for each of these grouping variables. The full conditionals for the μ parameters are, for the current value of k:

$$\mu|k, \ldots \sim N\left(\left(1^T \Psi_R^{-1} 1 + \tau_\mu\right)^{-1} 1^T \Psi_R^{-1} R, \left(1^T \Psi_R^{-1} 1 + \tau_\mu\right)^{-1}\right),$$

where $\Psi_R = \tau^{-1}I + (T\tau_{\phi_k})^{-1}M_k M_k^T$, where the M_k matrix is used to map each observation to their corresponding k group information. At the sub-terminal node level we have:

$$\phi_k|k,\ldots \sim N\left(\left(\tau M_k^T M_k + T\tau_{\phi_k} I\right)^{-1}(\tau M_k^T R + T\tau_{\phi_k}\mu), \left(\tau M_k^T M_k + T\tau_{\phi_k} I\right)^{-1}\right),$$

where now τ_{ϕ_k} is the value of the precision parameter for each grouping variable k, $k = 1,\ldots,K$. The prior distributions for $\tau_{\phi_k} \sim Ga(a_k, b_k)$ are set in a similar way as that of HEBART, where a simple intercept-only LME model is used to find the hyperparameter values using all grouping variables in a crossed model. We thus estimate $\hat{\tau}_{\phi_k}^{\text{LME}}$, extract the variance of this estimate and write it as $\hat{\sigma}^2(\hat{\tau}_{\phi_k}^{\text{LME}})$. The mean and variance are then used to provide point estimates of a_k and b_k for each grouping variable using standard method of moments.

To set the prior for $\tau \sim \text{Ga}(\nu/2, \lambda\nu/2)$, the overall precision parameter, we calibrate the values of ν and λ using the same rule as BART, but via the residual variance of the above LME model that includes all grouping variables. This strategy aims to yield a high probability that τ is bigger than $\hat{\tau}^{\text{LME}}$, so we find prior hyperparameters such that $P(\tau > \hat{\tau}^{\text{LME}}) = 0.95$. The complete steps for fitting this model are found in Algorithm 3.

5.2.2 Nested Random Effects HEBART (NHEBART)

We next extend the HEBART model so that, for a single tree, we have both terminal node predictions μ_j with prior $\mu_j \sim N(0, \tau_\mu^{-1})$, sub-terminal node predictions ϕ_{jk} with $\phi_{jk} \sim N(\mu, \tau_{\phi_k}^{-1}/P)$, and the extra sub-sub terminal node predictions γ_{jkl} with $\gamma_{jkl} \sim N(\phi_{jk}, \tau_\gamma^{-1}/P)$, which account for nested random effects (or grouping variables). See Figure 5.1 for a schematic as to how these are operationally used in the model structure, where we have an example tree of this model in the panel (c). Note that we have an extra level in each of the terminal nodes, and that those sub-levels can have a different number of classes. The indices here represent terminal node $j = 1,\ldots,b$, sub-terminal node group $k = 1,\ldots,g$, and sub-sub terminal node group $l = 1,\ldots,s$. In the developments below we vectorise each of

Algorithm 3 Crossed-RE HEBART Algorithm

Type: Metropolis within GIBBS for a Crossed-RE HEBART model

Require: y, X, grouping variables $z_k, k = 1, \ldots, K$

Ensure: Posterior distribution of trees $\mathcal{T}$, μ, ϕ, τ and τ_ϕ Initial values for α, β, σ_ϕ, τ, total number of trees P, number of grouping variables K, stumps $\mathcal{T}_1, \ldots, \mathcal{T}_P$, number of observations N, number of MCMC iterations $\mathcal{I}$ = burn-in+ post-burn-in, initial residual set $\mathbf{R}^{(1)} = \mathbf{y}$

for $i \leftarrow 1$ to $\mathcal{I}$ **do**

for $k \leftarrow 1$ to K **do**

Set $P_k = *\frac{P}{K}$

for $p_k \leftarrow 1$ to P_k **do**

1. Grow a new tree $\mathcal{T}^*_{p_k}$ tree by either growing, pruning, changing or swapping a root node
2. Calculate $\alpha(\mathcal{T}^*_{p_k}, \mathcal{T}_{p_k}) = \min\left\{1, \frac{P(R^{(i)}_p|\mathcal{T}^*_{p_k},\tau)P(\mathcal{T}^*_{p_k})}{P(R^{(i)}_{p_k}|\mathcal{T}_{p_k},\tau)P(\mathcal{T}_{p_k})}\right\}$;
3. Sample $u \sim U(0, 1]$
4. **if** $u < \alpha(\mathcal{T}^*_{p_k}, \mathcal{T}_{p_k})$ **then do** $\mathcal{T}_{p_k} = \mathcal{T}^*_{p_k}$

for $b \leftarrow 1$ to b_{p_k} **do**: Sample μ_{b,p_k}

for $j \leftarrow 1$ to $J_{b_{p_k}}$ **do**: Sample $\phi_{p_k,b,j}$

end for

end for

Update $\mathbf{R}^{(i)}_{p_k} = \mathbf{y} - \sum_{k=1}^{K}\sum_{t\neq p_k}^{P_k} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_{k,i}}, \mathcal{T}_{k,t}, \Theta_t)$

Update $\hat{f}_{ij} = \sum_{p_k=1}^{P_k} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_i}, \mathcal{T}_{p_k}, \Theta_{p_k})$

Sample τ_{ϕ_k}:

Sample a value $d \sim N(0, \sigma^2_{\tau_\phi})$ and make $\tau^*_{\phi_k} = \tau_{\phi_k} + d$

Calculate $\alpha(\tau^*_\phi, \tau_{\phi_k}) = \min\left\{1, \frac{P(\mathbf{y}|\tau^*_{\phi_k},\Theta)P(\tau^*_{\phi_k})\Phi(\tau_{\phi_k})}{P(\mathbf{y}|\tau_{\phi_k},\Theta)P(\tau_{\phi_k})\Phi(\tau^*_{\phi_k})}\right\}$;

Sample $u \sim U(0, 1]$

if $u < \alpha(\tau^*_{\phi_k}, \tau_{\phi_k})$ **then do** $\tau_\phi = \tau^*_{\phi_k}$

end for

Sample τ

end for

these terms and focus on a single terminal node (so that e.g. μ is a scalar) in a single tree. To update the trees and the parameters we use the partial residuals calculated as:

$$R = y - S^{(-t)}\gamma \tag{5.6}$$

where S is a binary matrix that allocates each observation in each tree to a sub-sub terminal node value in γ. The γ object now represents the stacked vector of all sub-sub-terminal node parameter values across all trees. $S^{(-t)}$ is the sub-matrix of S where the columns associated with the tree being updated are set to zero, such that when the residuals are calculated in the fitting algorithm, the predictions for the current tree are nullified and do not affect the sampling.

As with all BART models, the first step is to determine the marginal (log) likelihood after integrating out the terminal node parameters. We first write

$$\mathbf{R} \sim N(\mathbf{M_1}\gamma, \tau^{-1}I), \tag{5.7}$$

where $\mathbf{R}$ now represents those partial residuals in a single terminal node. M_1 is the sub-matrix of S that contains only those columns and rows for the current tree in the current terminal node. We abuse the above notation slightly to use γ from hereon as the vector of sub-sub terminal node values for an individual terminal node. When combined with the prior distributions above we can obtain a full conditional distribution to update γ (shown later) and marginalise Equation 5.7 over γ, to give:

$$\mathbf{R} \sim N(\mathbf{M_2}\phi,\ \tau^{-1}I + (P\tau_\gamma)^{-1}\mathbf{M_1}\mathbf{M_1}^T), \tag{5.8}$$

which now depends on ϕ. Further marginalisation yields:

$$\mathbf{R} \sim N(\mu 1,\ \tau^{-1}I + (P\tau_\gamma)^{-1}\mathbf{M_1}\mathbf{M_1}^T + (P\tau_\phi)^{-1}\mathbf{M_2}\mathbf{M_2}^T)), \tag{5.9}$$

where now $\mathbf{M}_2$ is the matrix that allocates each partial residual to its sub-terminal node grouping value. From this equation we define $\Delta_R = \tau^{-1}I + (P\tau_\gamma)^{-1}\mathbf{M_1}\mathbf{M_1}^T$ and $\Phi_R = \tau^{-1}I + (P\tau_\gamma)^{-1}\mathbf{M_1}\mathbf{M_1}^T + (P\tau_\phi)^{-1}\mathbf{M_2}\mathbf{M_2}^T)$. Both of these provide full

conditional distributions for all the node parameters, meaning γ, ϕ and μ, which can be written as:

$$\mu|\ldots \sim N((\mathbf{1}^T\Phi_R^{-1}\mathbf{1}+\tau_\mu)^{-1}(\mathbf{1}^T\Phi_R^{-1}\mathbf{R}),\ \ (\mathbf{1}^T\Phi_R^{-1}\mathbf{1}+\tau_\mu)^{-1}), \tag{5.10}$$

$$\phi|\ldots \sim N((\mathbf{M_2}^T\Delta_R^{-1}\mathbf{M_2}+(P\tau_\phi)\mathbf{I})^{-1}(\mathbf{M_2}^T\Delta_R^{-1}\mathbf{R}+(P\tau_\phi)\mu\mathbf{1}),\ \ (\mathbf{M_2}^T\Delta_R^{-1}\mathbf{M_2}+(P\tau_\phi)\mathbf{I})^{-1}), \tag{5.11}$$

$$\gamma|\ldots \sim N((\tau\mathbf{M_1}^T\mathbf{M_1}+\tau_\gamma P\mathbf{I})^{-1}(\tau\mathbf{M_1}\mathbf{R}+\tau_\gamma P\phi\mathbf{1}),\ \ (\tau\mathbf{M_1}^T\mathbf{M_1}+\tau_\gamma P\mathbf{I})^{-1}). \tag{5.12}$$

The final marginalisation of the posterior distribution of the residuals gives us:

$$\mathbf{R} \sim N(0,\ \ \tau^{-1}I+(P\tau_\gamma)^{-1}\mathbf{M_1}\mathbf{M_1}^T+(P\tau_\phi)^{-1}\mathbf{M_2}\mathbf{M_2}^T+\tau_\mu\mathbf{1}\mathbf{1}^T) \tag{5.13}$$

which is used for updating the individual trees as in the Metropolis-Hastings ratio part of the fitting algorithm.

For the two sub-node precision parameters τ_ϕ and τ_γ, we use a random-walk update that provides reliable samples from their posteriors (Brooks et al., 2011b). A standard Normal is used to create the proposal values τ_ϕ^* and τ_γ^*, which are either accepted or rejected if their likelihood is higher than the previous value. For τ_ϕ^*, this decision is based on the acceptance probability, $\alpha(\tau_\phi^*, \tau_\phi) = \min\left\{1, \frac{P(\mathbf{y}|\tau_\phi^*,\Theta)P(\tau_\phi^*)\Phi(\tau_\phi)}{P(\mathbf{y}|\tau_\phi,\Theta)P(\tau_\phi)\Phi(\tau_\phi^*)}\right\}$, where $\Phi(x)$ represents the CDF of the standard Normal, and analogously for τ_γ and τ_γ^*. Since we are sampling from a distribution with limited support (positive values only, as those are precision parameters), this sampler is used to account and correct for that. Lastly, for updating τ we can use the main likelihood term:

$$y \sim N(\mathbf{S}\gamma, \tau^{-1}I)$$

and combine this with the standard prior on $\tau \sim Ga(a, b)$ to give a full closed-form conditional update. The full algorithm for the NHEBART model is given in Algorithm 4.

5.3 Applications & Results

5.3.1 Crossed Random Effects HEBART

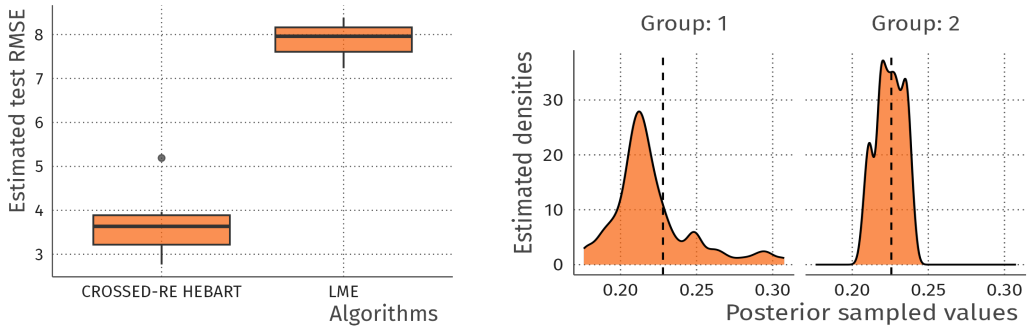
We first experiment with CHEBART on a simulated data scenario, where the response variable y is simulated as a sum of two tree structures and two grouping variables, so that each tree corresponds to a single grouping variable. The first tree (grouping variable) has three different levels, with its allocation to each of the simulated values being random, with a different set of parameters for each simulated node in the tree. The second tree (grouping variable) has five levels also allocated at random, and its corresponding sets of parameters for the tree nodes. There are in total two covariates used to create the tree structures, each simulated from a $U(0,1)$ distribution. By splitting the simulated covariates X_1, X_2 at random points, we sample $n = 500$ values as

$$y = G(X_1, Z_1, T_1, \Theta_1) + G(X_2, Z_2, T_2, \Theta_2) + \epsilon \tag{5.14}$$

where $\epsilon \sim N(0,1)$. Tree T_1 consists of three terminal nodes with the partitions $I(X_1 < 0.5)$, $I(X_1 < 0.75)$, $I(X_1 \geq 0.75)$ respectively, and having terminal node values $\mu_1 = -5, \mu_2 = 10, \mu_3 = 0$. The sub-groups of tree 1 are: $\phi_1 = \{-7, -5, -3\}$, $\phi_2 = \{8, 10, 12\}$, and $\phi_3 = \{-2, 0, 2\}$. For tree T_2 the partitions are $I(X_2 < 0.4)$, $I(X_2 < 0.85)$, $I(X_2 \geq 0.85)$ having terminal node values $\mu_1 = 8, \mu_2 = -3, \mu_3 = 0$, and sub-groups $\phi_1 = \{8, 15, 7, 4, -1\}$, $\phi_2 = \{7, -2, -10, 2, 5\}$, and $\phi_3 = \{-2, 0, 2, 20, 4\}$. This experiment uses 10 different train and test sets as our cross-validation setting (Refaeilzadeh et al., 2009). To these simulated sets, we apply our proposed algorithm and compare it to the corresponding LME model with two random effects, always fitting the algorithms on the 10 training sets and making predictions for the 10 testing sets.

In Figure 5.2, we can see that Crossed-RE HEBART produces the smallest root mean squared errors for the test sets, as per the boxplot shown. The highest RMSE value produced by Crossed-RE HEBART is smaller than the lowest RMSE value produced by LME, showing its much more powerful prediction capacity for this simulated data, even though our model's RMSE varies slightly more. We also

provide plots for the posterior samples of the σ parameters for the two grouping variables, where we can see that the average values are very close to the true value of 0.20. In addition, Figure 5.3 shows the convergence status for the two parameters: no absurd convergence problems have been observed. We now move on to fitting our model to a real data set.



(a) Test RMSE for CHEBART and LME

(b) Sampled values from the posterior distributions of σ_{τ_1} and σ_{τ_2}

Figure 5.2: Simulation performance of CHEBART predictions and posterior sampled values. In (a) we have the boxplots of the 10 test RMSEs for the CHEBART and LME algorithms, with CHEBART performing better. In (b) we can see the posterior sampled values for the σ parameters of the two grouping variables, with their corresponding averages, which are both close to the correct 0.20 value.

For our real-world data evaluation we apply our model to the widely-used Gapminder data (Lang, 2011). For our experiment, the data consists of the life expectancy values (in years) for 20 different countries from 1950 to 2018. We use life expectancy as the response. For the covariates we use: year; an indicator of the percentage of the time passed; and the number of years passed. We use country and a categorical version of the year variable (each 10 years/decade to account for a decade effect) as the two grouping variables. Again we split this dataset into 10 different training and test sets, where the testing set is composed of all the observations for 15 sampled years (for all countries), and the corresponding training set is composed of the observations for the remaining years. Fully removing a set of years from the training set makes it harder for the model to predict for such years, since it has no information of what happened in the removed years. We also

compare our model against an LME with the two groups as random effects both in the slope and in the intercept, to allow for as much flexibility as is possible for LME, our main competitor.

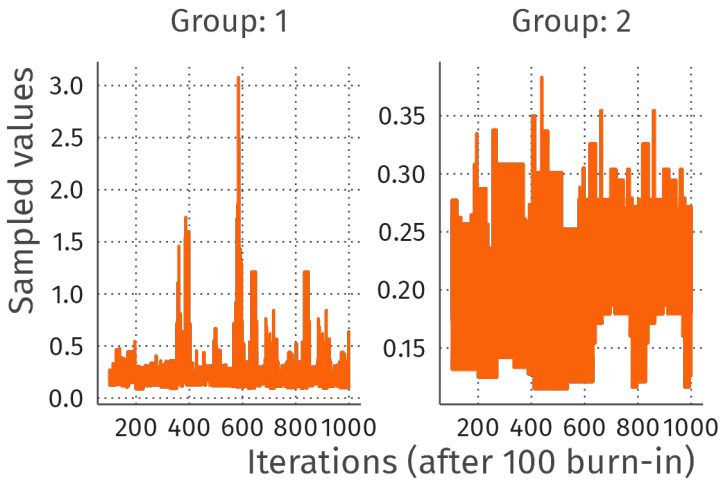


Figure 5.3: Convergence plots of posterior distributions of σ_{τ_1} and σ_{τ_2}, indicating good convergence but which could be improved.

Figure 5.4 shows the results for this experiment as the predictions for six different countries using CHEBART and LME. The CHEBART predictions are very close to the actual values due to the increase in flexibility of the predictions from the non-parametric structure and the presence of the two grouping variables. The LME algorithm clearly cannot adapt as well to the wiggly behavior of the observations for some of the groups/countries, such as South Africa and Turkey. Even for more linear situations, as for the United States, LME performs worse than our model, which is probably a result of the higher prediction capacity of a tree-based model in comparison to a linear equation. In addition, our model also has the advantage of being able to predict when grouping variable information is missing, e.g., when we have the year information but a decade is not complete yet, while still keeping the tree-based model flexibility and prediction power. The average test RMSEs for the two models confirm the evidence from the predictions, as there is no intersection between the empirical confidence intervals for the RMSE averages,

showing that we do indeed have a big advantage over LME, even when random slopes and intercepts are allowed.

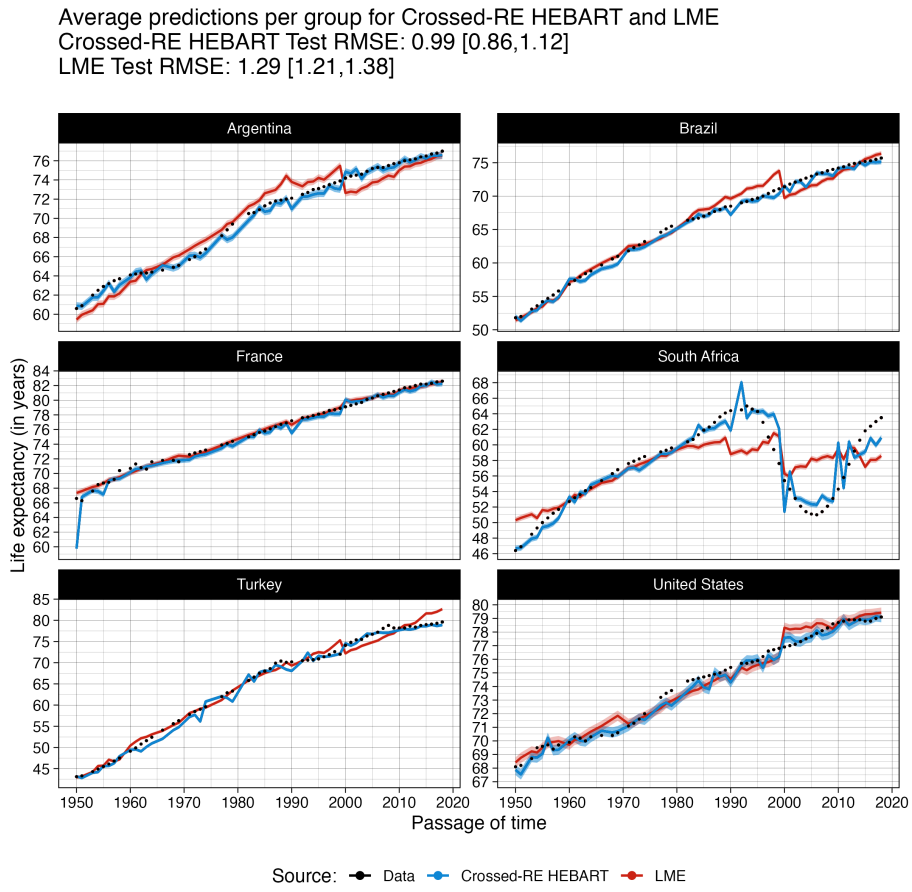


Figure 5.4: Predictions for six countries of the gapminder data, with the out-of-sample LME predictions in red and the out-of-sample CHEBART predictions in blue. The CHEBART predictions are closer to the actual values due to the increase in flexibility of the predictions from the non-parametric structure and the presence of the two grouping variables. LME does not capture well the non-linearities in the observations.

5.3.2 Nested Random Effects HEBART

For this algorithm, we also start with a simple simulation example, though more challenging than the CHEBART example above. In this case, the response variable y is simulated as a sum of two tree structures with two nested groups with

their respective parameters. In each terminal node of each tree the first grouping variable has four different levels, and each group has a different number of sub-groups (3, 2, 5, and 3 sub-groups respectively). Each group and sub-group have their own simulated parameters, and are used to create the response variable y in combination with the tree structures. There are also two covariates, each simulated from standard Normal distribution, that are split at some certain points to create the tree structures in the data. With this, we sample $n = 500$ values as:

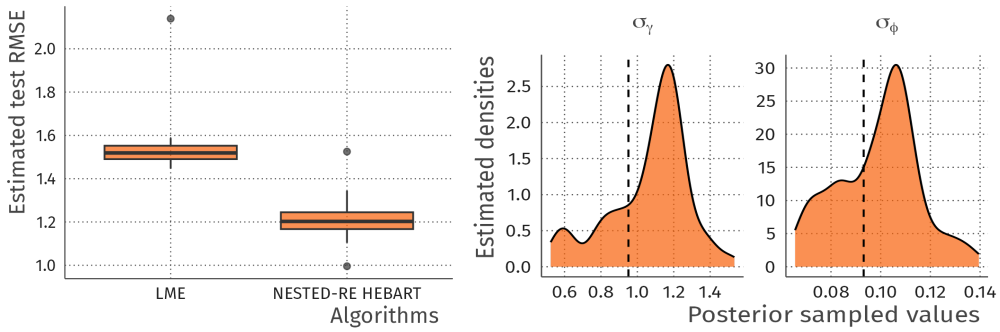
$$y = G(X_1, Z_1, T_1, \Theta_1) + G(X_2, Z_2, T_2, \Theta_2) + \epsilon \tag{5.15}$$

where $\epsilon \sim N(0, 1)$. Tree structure T_1 has three terminal nodes created as $I(X_1 < 0, X_2 < 0)$, $I(X_1 < 0, X_2 \geq 0)$, and $I(X_1 \geq 0)$ respectively. Tree structure T_2 has the single split created from $I(X_2 < 0)$ vs $I(X_2 \geq 0)$. For each terminal node in each tree we simulate $\mu \sim N(0, \tau_\mu^{-1} = 3^{-1})$. For each of the 4 groups at the sub-terminal node level we simulate $\phi \sim N(0, \tau_\phi^{-1} = 3^{-1})$. For each of the sub-groups we simulate $\lambda \sim N(0, \tau_\lambda^{-1} = 2^{-1})$. Recall that the number of λ sub-terminal node values will vary between groups.

This experiment uses 10 different train and test sets, randomly sampled in the 75%/25% proportion. We apply our Nested-RE HEBART algorithm and compare it to the corresponding LME model, fitting the algorithms on the 10 training sets and making predictions for the 10 testing sets for evaluating the results on out-of-sample data.

Figure 5.5 has the resulting boxplots for the test RMSEs of both models. From the image, we can clearly see the NHEBART model's advantage over LME, as the RMSEs are much lower for all simulated data sets. On the right side of the Figure, we have the empirical distribution of the posterior sampled values of σ_ϕ and σ_γ, representing the standard deviations of the group and sub-group parameters. Both distributions have a clear center and as per the low RMSEs, we have good evidence that the dispersion parameters are being correctly estimated.

As a second example, we continue using the previously mentioned Gapminder data (Lang, 2011), on a very similar fashion to what was used for the previous algo-



(a) Test RMSE for NHEBART and LME

(b) Sampled values from the posterior distributions of σ_ϕ and σ_γ

Figure 5.5: Simulation performance of NHEBART predictions and posterior sampled values. In (a) we have the boxplots of the 10 test RMSEs for the NHEBART and LME algorithms, with NHEBART performing much better. In (b) we can have the posterior sampled values for the σ_ϕ and σ_γ, corresponding to the group and sub-group variables. For both grouping levels there is a clear center on the empirical distributions.

rithm (CHEBART). For this experiment, the data consists of the life expectancy values (in years) for 20 different countries from 1950 to 2018, where we filtered out continents that had only one country available, as those will be our group and sub-group variables, which present a natural nested data structure. Life expectancy continues to be the response, and we use year (time passage) and and the number of years passed as the two covariates. We split the dataset into 10 different training and test sets, where the testing set is composed of all the observations for 15 sampled years (for all countries), and the corresponding training set is composed of the observations for the remaining years (similar to the previous experiment presented here). The comparison is made against LME with nested random effects, namely the continent and country variables present in the data, which corresponds to the model we are fitting when using Nested-RE HEBART.

Figure 5.6 shows the results for this experiment as the predictions for six different countries using NHEBART and LME. Once again, our predictions are closer to reality than that of LME, even with the nested random effects and varying intercept

and slopes. NHEBART adapts to the data for most of the cases shown, while LME fails to capture relationships that are even fairly linear, such as the data for the United States. In addition, a few model specification issues were reached while using LME, which did not allow two random slopes to be used because of the colinearity between the covariates. This is something that would not happen with HEBART, though more investigation on feature selection and avoiding the use of correlated variables is needed. In comparison to the CHEBART results (Figure 5.4, we have a bigger RMSE for NHEBART, but those are quite different models. Nevertheless, it is possible to infer that using the categorical year variable makes a big difference for the predictions. A combination of the two algorithms, mixing the nested random effects with the usage of multiple groups accounting for the decade effect would probably be ideal for this data, which can also be included in further work.

Lastly, in Figure 5.7 we have the predictions using only the first group-level parameters, meaning we make predictions for the continents without using the country information. This represents one more advantage of our model, that we do not need to have knowledge about both the continent and the country to make predictions. In case the second variable is missing, we can still use the continent-level parameters to create the predicted values. In this case, we still have a better performance than the corresponding LME model, as our predictions can approximate the data in a more accurate way.

5.3.3 Code & Data Availability

The package created to run the Crossed-RE HEBART algorithm is written in the language `R` (R Core Team, 2018), and available at `https://github.com/brunaw/MHEBART`. The package for the Nested-RE model is available at `https://github.com/brunaw/NHEBART`. To complement it, all the experiments code & data are available in a third repository at the URL `https://github.com/brunaw/hebart-experiments`. In the future, we plan to have a single optimized `R` package that contemplates all HEBART options currently available.

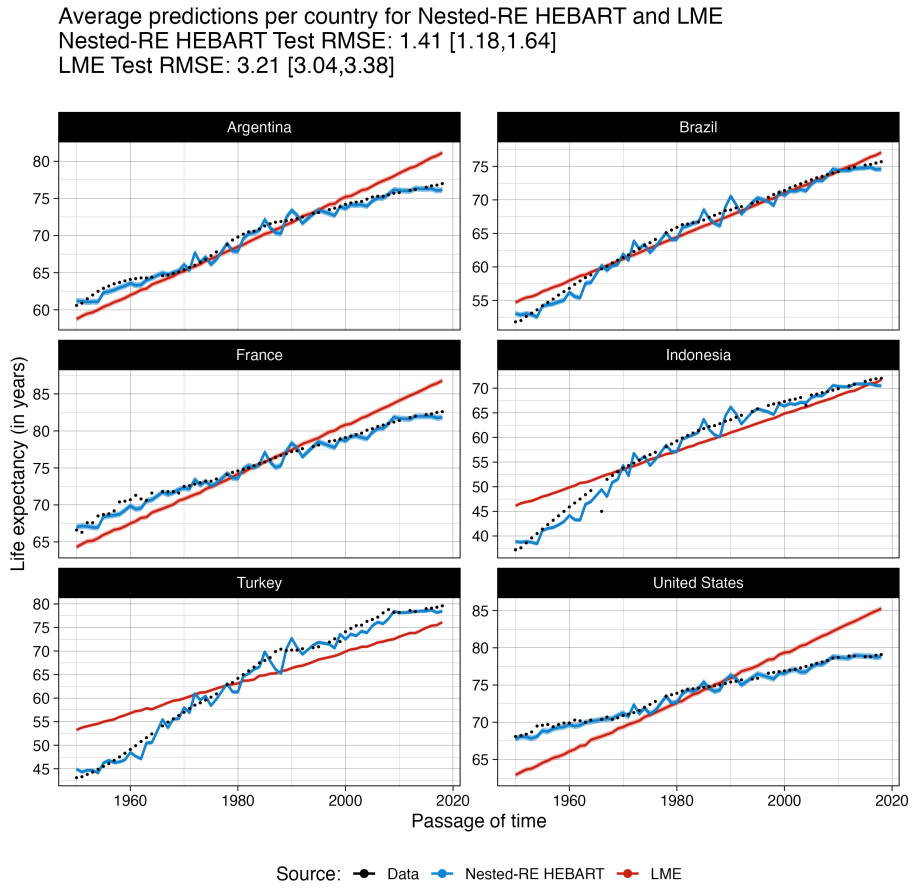


Figure 5.6: Predictions for six countries of the gapminder data, with the out-of-sample LME predictions in red and the out-of-sample NHEBART predictions in blue. The NHEBART predictions are much closer to the actual values, and LME struggles to properly model the observations.

5.3.4 Conclusions

The two proposed extensions to HEBART now provide a complete suite of algorithms whereby many of the most common linear mixed-effects models can be replaced with HEBART versions. We have described and outlined two important extensions to HEBART:

- the Crossed random effects HEBART, which deals with multiple categorical grouping variables in the same model;

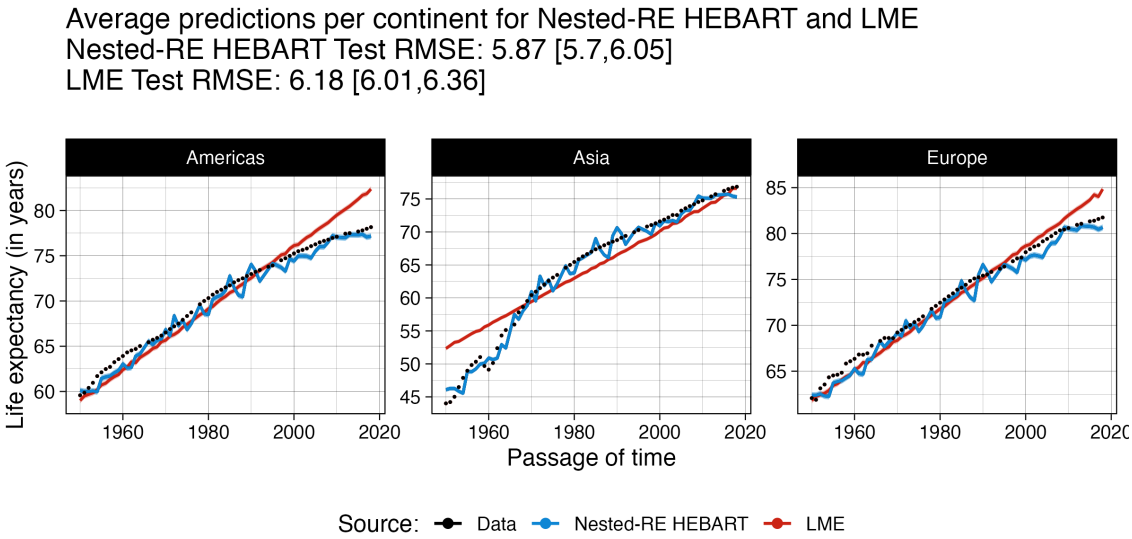


Figure 5.7: Predictions for continents (instead of countries) of the gapminder data, with the out-of-sample LME predictions in red and the out-of-sample NHEBART predictions in blue. On a continent level, our predictions are still better than LME, as our model is more able to capture the complicated relationship between time and life expectancy in years.

- the Nested random effects HEBART which accounts for situations where we have nested multiple grouping variables

The first model is useful in situations where there are more than one grouping variable known to be affecting the data-generating process, and that we need to account for that when modeling the response. This algorithm uses a proportional number of trees with each grouping variable, allowing them to have their effect captured in the estimation. Here, we have verified through experiments using simulated and real datasets that, in terms of producing accurate predictions, the Crossed random effects HEBART is able to beat standard Crossed-RE algorithms, such as the classic LME model. In addition to that, our model is also able to consistently find full posterior distributions to both the grouping and terminal node parameters, enjoying the properties of Bayesian algorithms that provide uncertainty intervals for all quantities estimated.

For the second proposed model, we also allow the presence of more than one grouping variable, but now in a nested rather than a combined fashion. The trees have not only one group sub-node, but a further sub-sub-node level, where the value of the second-level groups depend on the first-level ones, thereby creating the nested structure. Such a nesting structure is found in many hierarchical data modeling problems (Gelman and Hill, 2006). Similar to Crossed HEBART, we also demonstrated the nested version of HEBART to be able to beat the corresponding LME model for both simulated and real datasets where the nested structure was an important component of the task.

Our current implementations are based on R code and allow for only a single nested layer in the trees. Future extensions would turn our code into a more fully featured package based on e.g. `Rcpp` (Eddelbuettel and François, 2011) to speed up model fitting. Such a package may further extend the work to include multiple nested layers and beyond simple regression problems into generalised linear models.

Algorithm 4 Nested-RE HEBART Algorithm

Type: Metropolis within GIBBS for a Nested-RE HEBART model

Require: y, X, nested grouping variables $z_k, k = 1, \ldots, K$

Ensure: Posterior distribution of trees $\mathcal{T}$, μ, ϕ, τ and τ_ϕ Initial values for α, β, σ_ϕ, τ, total number of trees P, number of grouping variables K, stumps $\mathcal{T}_1, \ldots, \mathcal{T}_P$, number of observations N, number of MCMC iterations $\mathcal{I}$ = burn-in+ post-burn-in, initial residual set $\mathbf{R}^{(1)} = \mathbf{y}$

for $i \leftarrow 1$ to $\mathcal{I}$ **do**

for $p \leftarrow 1$ to P **do**

1. Grow a new tree $\mathcal{T}_p^*$ tree by either growing, pruning, changing or swapping a root node
2. Calculate $\alpha(\mathcal{T}_p^*, \mathcal{T}_p) = \min\left\{1, \frac{P(R_p^{(i)}|\mathcal{T}_p^*,\tau)P(\mathcal{T}_p^*)}{P(R_p^{(i)}|\mathcal{T}_p,\tau)P(\mathcal{T}_p)}\right\}$;
3. Sample $u \sim U(0,1]$
4. **if** $u < \alpha(\mathcal{T}_p^*, \mathcal{T}_p)$ **then do** $\mathcal{T}_p = \mathcal{T}_p^*$

for $b \leftarrow 1$ to b_p **do**: Sample $\mu_{b,p}$

for $j \leftarrow 1$ to J_{b_p} **do**: Sample $\phi_{p,b,j}$

for $k \leftarrow 1$ to $K_{b_p,j}$ **do**: Sample $\gamma_{p,b,j,k}$

end for

end for

end for

Update $\mathbf{R}_p^{(i)} = \mathbf{y} - \sum_{t\neq p}^{P} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_i}, \mathcal{T}_t, \Theta_t)$

Update $\hat{f}_{ij} = \sum_{p_k=1}^{P_k} \mathbf{G}(\mathbf{X_{ij}}, \mathbf{z_i}, \mathcal{T}_{p_k}, \Theta_{p_k})$

Sample τ_ϕ:

Sample a value $d \sim N(0, \sigma^2_{\tau_\phi})$ and make $\tau^*_{\phi_k} = \tau_\phi + d$

Calculate $\alpha(\tau^*_\phi, \tau_{\phi_k}) = \min\left\{1, \frac{P(\mathbf{y}|\tau^*_\phi,\Theta)P(\tau^*_\phi)\Phi(\tau_\phi)}{P(\mathbf{y}|\tau_\phi,\Theta)P(\tau_{\phi_k})\Phi(\tau^*_\phi)}\right\}$;

Sample $u \sim U(0,1]$

if $u < \alpha(\tau^*_\phi, \tau_\phi)$ **then do** $\tau_\phi = \tau^*_\phi$

Sample τ_γ:

Sample a value $d \sim N(0, \sigma^2_{\tau_\gamma})$ and make $\tau^*_{\gamma_k} = \tau_\gamma + d$

Calculate $\alpha(\tau^*_\gamma, \tau_{\gamma_k}) = \min\left\{1, \frac{P(\mathbf{y}|\tau^*_\gamma,\Theta)P(\tau^*_\gamma)\Phi(\tau_\gamma)}{P(\mathbf{y}|\tau_\gamma,\Theta)P(\tau_{\gamma_k})\Phi(\tau^*_\gamma)}\right\}$;

Sample $u \sim U(0,1]$

if $u < \alpha(\tau^*_\gamma, \tau_\gamma)$ **then do** $\tau_\gamma = \tau^*_\gamma$

Sample τ

end for

CHAPTER 6

Conclusions

This thesis proposes a few extensions to overcome key limitations of both Bayesian Additive Regression Trees and standard tree-based models. Such statistical models are, in general, highly flexible as they can deal with complex interactions between the covariates, they are consistent, fairly easy to understand and able to create accurate predictions for many sorts of datasets and machine learning tasks. This is enough to justify the need for tackling and coming up with solutions to some major bottlenecks of the tree-based models, as the lack of efficient ways to select the most important features and proper extensions to deal with grouped data, which are part of an important ramification of statistical methods. In this final Chapter, we briefly discuss the presented methods and some of their main limitations and potential future work.

The final conclusion shows the similarities across HEBART, NHEBART, and CHEBART by focusing on their core structure as members of an "extended BART family," united by the common framework of embedding hierarchical or grouped data structures into the BART model while allowing flexibility at the terminal node level. Essentially, each variant introduces a design matrix tailored to the specific grouping structure—be it hierarchical, nested, or crossed—which defines the partitioning and interaction handling within the terminal nodes. This design enables all these models to achieve more granular and accurate predictions, accounting

for grouped data in a non-parametric way. In addition to highlighting the shared lineage, we note that these models address practical limitations of BART, such as dealing with complex random effects and feature selection in hierarchical contexts. They improve upon traditional BART by integrating grouping information seamlessly, allowing for context-specific predictions while retaining BART's predictive power and uncertainty calibration.

In this document, in Chapter 3 we first propose a method that generalizes feature selection via gain penalization for standard tree-based models. The presented technique is based on a new gain penalization idea that exhibits a general local-global regularization for tree models, where we create a generic penalization function composed of a mixture of a baseline parameter and of a covariate-based function. With this, the new method allows for full flexibility in the choice of feature-specific importance weights, while also applying a global penalization to all of the features. In our proposal, we discuss a few possible covariate-based functions that can be used to create the penalization values and their main implications/consequences. We also study and extensively explain other aspects of the method, such as the effect of different values for each parameter and the best ways to tune these values, as well as the best usage of the depth hyper-parameter, an extension that allows our method also to control the penalization via the current depth of the tree being fit. As a result, we have a flexible and scalable gain penalization method that can best select the most important features, even in extreme cases where we have several highly correlated or useless features. We validate our method on both simulated and real datasets with the implementation we provided as an addition to the the popular and efficient `R` package `ranger`.

In the following, Chapter 4 changes the subject a bit and switches to an extension of the Bayesian Additive Regression Trees algorithm, which is a more specific Bayesian tree-based model. We propose a simple yet powerful extension of BART, named Hierarchical Embedded BART (HEBART) for its hierarchical structure and the addition of the grouping variable as a fundamental part of the algorithm. In other words, HEBART is to BART what an LME model is for a linear regression. Our model allows for random effects to be included at the terminal node level of the BART regression trees, making HEBART a powerful non-parametric

alternative to mixed effects models. One of the main advantages of our model is that we avoid the need for the user to specify the structure of the random effects, but we also maintain the prediction and uncertainty calibration properties of standard BART. This allows us to have a more specific prediction for situations where the grouping values are known, but also for when we do not have such information, which keeps the famous prediction power of BART. Using a few simulated and real-world examples, we demonstrate how this new extension yields superior predictions for some of the standard mixed effects models' example data sets, and yet still provides consistent estimates of the random effect variances. With our mathematical model, it also comes the computational implementation, which we once again have done via the statistical language `R` and made it available at `https://github.com/brunaw/hebartBase`, where the main package is stored, and and `https://github.com/brunaw/hebart-experiments` contains the code to replicate all the examples, with their corresponding auxiliary files.

While creating HEBART, we also identified a few possible extensions to it, which are even mentioned in the text as the future work. This motivated the methods proposed in Chapter 5, where we describe and exemplify two important extensions to HEBART: I) the Crossed Random Effects HEBART, which deals with multiple random effects in the same model; the II) the Nested Random Effects HEBART, that accounts for situations where we have more than one random effect, but they come on a nested fashion. These extensions are similar to what LME already has, as those are common modelling situations that need to be accounted for when trying to create generic models for hierarchical data. The first model is useful in situations where there are more than one grouping variable known to be affecting the data-generating process, and that we need to consider when modeling the response. For this algorithm, a proportional number of trees is used with each grouping variable used, allowing them to have their effect captured in the estimation. As for the second model, we also have the presence of more than one grouping variable, but now in a nested rather than a combined fashion. The trees used in this model have not only one group subnode, but two subnode levels, where the value of the second-level groups depend on the first-level ones, allowing for much more flexibility and accurate predictions. We experiment with

both models on simulated and real-world datasets, and, with this, we demonstrate their prediction capacity and potentiality to be powerful machine learning models for hierarchical data. The computational implementation is done via the statistical language `R` and it is available at `https://github.com/brunaw`, and and `https://github.com/brunaw/hebart-experiments` has the examples shown in the text. In the future, we intend to have one single, optimized package with all HEBART extensions, in a fashion similar to the `lme4` package interface.

To conclude, the methods proposed here cover a wide range of issues that existed for tree-based models. The methods presented are innovative and tackle actual and practical problems, representing important extensions to tree models in general, which will likely have a large impact on how people deal with the beforementioned issues, as new tools are now available to solve them. With this, it also comes the possibility of keep extending the proposed models, in a continuous attempt to achieve better results. For instance, Chapter 3 can be extended in terms of finding theoretical properties of our gain penalization approaches, as well as parameter optimization (with, e.g. Bayesian Optimization, Snoek et al. (2012)), and conducting a wider comparison of approaches with a similar context (Johnson and Zhang (2013); Nan and Saligrama (2017); Nan et al. (2016), for example). As for HEBART, beyond our own extensions proposed in Chapter 5, we also see potentiality in explicitly modeling joint random effects using the multivariate normal distribution and, consequently, estimating covariances between grouping variables, as it is fair to assume that multiple grouping variables might be correlated in some certain situations. In addition, merging our ideas to other recent (yet, different) BART extensions such as such as SOFT-BART (Linero, 2018a) and MOTR-BART (Prado et al., 2021a) can substantially enhance the predictive capabilities our model, helping BART achieve even more visibility and usage as a powerful machine learning model that can be applied to many different prediction scenarios.

In sum, these extensions collectively demonstrate the versatility and adaptability of the BART framework when appropriately modified to handle hierarchical, nested, and crossed data structures. This not only shows the flexibility of BART as a machine learning model but also illustrates the power of thoughtful adaptation in

addressing domain-specific challenges in grouped data modeling. Looking forward, continued development and unification of these models into a single optimized package would streamline their usability and make these advanced modeling tools even more accessible.

Bibliography

Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2008). Bayesian additive regression trees-based spam detection for enhanced email privacy. In *2008 Third International Conference on Availability, Reliability and Security*, pages 1044–1051. 3

Alexander, D. C., Zikic, D., Zhang, J., Zhang, H., and Criminisi, A. (2014). Image quality transfer via random forest regression: applications in diffusion mri. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014: 17th International Conference, Boston, MA, USA, September 14-18, 2014, Proceedings, Part III 17*, pages 225–232. Springer. 11

Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503. 39

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750. 39

Anderson, E. (1936). The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509. 12

Bates, D., Maechler, M., Bolker, B., Walker, S., Christensen, R. H. B., Singmann, H., Dai, B., Scheipl, F., and Grothendieck, G. (2011). Package 'lme4'. *Linear*

mixed-effects models using S4 classes. R package version, 1(6). 43, 50, 54, 56, 64

Belenky, G., Wesensten, N. J., Thorne, D. R., Thomas, M. L., Sing, H. C., Redmond, D. P., Russo, M. B., and Balkin, T. J. (2003). Patterns of performance degradation and restoration during sleep restriction and subsequent recovery: A sleep dose-response study. *Journal of sleep research*, 12(1):1–12. 56

Belgiu, M. and Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS journal of photogrammetry and remote sensing*, 114:24–31. 11

Blattenberger, G. and Fowles, R. (2014). Avalanche forecasting: Using bayesian additive regression trees (bart). In *Demand for Communications Services–Insights and Perspectives*, pages 211–227. Springer. 18, 68

Bonato, V., Baladandayuthapani, V., Broom, B. M., Sulman, E. P., Aldape, K. D., and Do, K.-A. (2011). Bayesian ensemble methods for survival prediction in gene expression data. *Bioinformatics*, 27(3):359–367. 3

Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140. 2

Breiman, L. (1996b). Bagging predictors. *Machine Learning*, 24(2):123–140. 11, 26

Breiman, L. (2001a). Random forests. *Machine learning*, 45(1):5–32. 1, 11

Breiman, L. (2001b). Random Forests. *Machine Learning*. 27

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis. 1, 7, 9, 11, 25

Brooks, S., Gelman, A., Jones, G., and Meng, X. (2011a). *Handbook of Markov Chain Monte Carlo*. CRC press, New York, USA. 3

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011b). *Handbook of markov chain monte carlo*. CRC press. 15, 19, 52, 78

Cano, G., Garcia-Rodriguez, J., Garcia-Garcia, A., Perez-Sanchez, H., Benediktsson, J. A., Thapa, A., and Barr, A. (2017). Automatic selection of molecular descriptors using random forest: Application to drug discovery. *Expert Systems with Applications*, 72:151–159. 11

Carlson, C. J. (2020). embarcadero: Species distribution modelling with bayesian additive regression trees in r. *Methods in Ecology and Evolution*, 11(7):850–858. 18, 68

Chen, P. and Popovich, P. (2002). *Correlation: parametric and nonparametric measures*. Number Nº 137-139 in Sage university papers series. no. 07-139. Sage Publications. 29

Chen, T., He, T., Benesty, M., Khotilovich, V., and Tang, Y. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4. 42

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948. 2, 14, 21, 45

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010a). BART: Bayesian additive regression trees. *Annals of Applied Statistics*. 31

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010b). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298. 15, 16, 43, 44, 45, 46, 47, 56, 64, 66, 67, 72

Clark, T. E., Huber, F., Koop, G., Marcellino, M., and Pfarrhofer, M. (2021). Tail forecasting with multivariate Bayesian additive regression trees. *Federal Reserve Bank of Cleveland, Working Paper No. 21-08*. 3

Cortez, P. (2016). *rminer: Data Mining Classification and Regression Methods*. R package version 1.4.2. 30

Cortez, P. and Embrechts, M. J. (2013). Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1 – 17. 30

Deng, H. (2013). Guided random forest in the rrf package. *arXiv preprint arXiv:1306.0237*. 35

Deng, H. and Runger, G. (2012). Feature selection via regularized trees. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. 13, 23, 24, 27, 38

Deng, H. and Runger, G. (2013). Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489. 5, 14, 24, 27

Denison, D. G., Mallick, B. K., and Smith, A. F. (1998). Bayesian MARS. *Statistics and Computing*, 8(4):337–346. 2

Deshpande, S. K., Bai, R., Balocchi, C., Starling, J. E., and Weiss, J. (2020). VCBART: Bayesian trees for varying coefficients. *arXiv preprint arXiv:2003.06416*. 3, 18, 68

Diaz-Uriarte, R. (2007). GeneSrF and varSelRF: a web-based tool and R package for gene selection and classification using random forest. 23, 24, 38, 39

Dorie, V. (2020). dbarts: Discrete Bayesian additive regression trees sampler. R package version 0.9-19. 54

Eddelbuettel, D. and François, R. (2011). Rcpp: Seamless r and c++ integration. *Journal of statistical software*, 40:1–18. 88

Friedman, J. and Popescu, B. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2. 23

Friedman, J. H. (1991). Rejoinder: Multivariate Adaptive Regression Splines. *The Annals of Statistics*. 34

Gelman, A. and Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press. 4, 43, 44, 47, 51, 69, 88

Goldstein, B. A., Hubbard, A. E., Cutler, A., and Barcellos, L. F. (2010). An application of random forests to a genome-wide association dataset: methodological considerations & new findings. *BMC genetics*, 11(1):1–13. 11

Goldstein, B. A., Polley, E. C., and Briggs, F. B. (2011). Random forests for genetic association studies. *Statistical applications in genetics and molecular biology*, 10(1). 11

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537. 39

Green, D. P. and Kern, H. L. (2012). Modeling heterogeneous treatment effects in survey experiments with Bayesian additive regression trees. *Public opinion quarterly*, 76(3):491–511. 3

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182. 23

Hahn, P. R., Murray, J. S., and Carvalho, C. M. (2020). Bayesian regression tree models for causal inference: regularization, confounding, and heterogeneous effects. *Bayesian Analysis*. 18, 68

Hastie, T. and Tibshirani, R. (2000). Bayesian backfitting (with comments and a rejoinder by the authors). *Statistical Science*, 15(3):196–223. 3, 19

Hastie, Trevor, Tibshirani, Robert, Friedman, J. (2009). *The Elements of Statistical Learning, Second Edition*. 30

He, J., Yalov, S., and Hahn, P. R. (2019). XBART: Accelerated Bayesian additive regression trees. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1130–1138. 3

Hernández, B., Raftery, A. E., Pennington, S. R., and Parnell, A. C. (2018). Bayesian additive regression trees using Bayesian model averaging. *Statistics and computing*, 28(4):869–890. 3, 18, 68

Hill, J. and Su, Y.-S. (2013). Assessing lack of common support in causal inference using Bayesian nonparametrics: Implications for evaluating the effect of breastfeeding on children's cognitive outcomes. *The Annals of Applied Statistics*, pages 1386–1420. 3

Hill, J. L. (2011). Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240. 3, 18, 68

Johnson, R. and Zhang, T. (2013). Learning nonlinear functions using regularized greedy forest. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):942–954. 42, 93

Kapelner, A. and Bleich, J. (2016). bartMachine: Machine learning with Bayesian additive regression trees. *Journal of Statistical Software, Articles*, 70(4):1–40. 21

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154. 42

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673. 39

Kindo, B. P., Wang, H., Hanson, T., and Peña, E. A. (2016a). Bayesian quantile additive regression trees. *arXiv preprint arXiv:1607.02676*. 3, 18, 64, 68

Kindo, B. P., Wang, H., and Peña, E. A. (2016b). Multinomial probit Bayesian additive regression trees. *Stat*, 5(1):119–131. 3, 18, 21, 64, 68

Lang, B. (2011). Gapminder: bringing statistics to life. *Teaching Geography*, 36(1):17. 58, 80, 83

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22. 41

Linero, A. R. (2017). A review of tree-based bayesian methods. *Communications for Statistical Applications and Methods*, 24(6). 18, 68

Linero, A. R. (2018a). Bayesian regression trees for high-dimensional prediction and variable selection. *Journal of the American Statistical Association*, 113(522):626–636. 18, 60, 68, 93

Linero, A. R. (2018b). Bayesian regression trees for high-dimensional prediction and variable selection. *Journal of the American Statistical Association*, 113(522):626–636. 20

Linero, A. R., Sinha, D., and Lipsitz, S. R. (2020a). Semiparametric mixed-scale models using shared Bayesian forests. *Biometrics*, 76(1):131–144. 3

Linero, A. R., Sinha, D., and Lipsitz, S. R. (2020b). Semiparametric mixed-scale models using shared bayesian forests. *Biometrics*, 76(1):131–144. 18, 68

Linero, A. R. and Yang, Y. (2018a). Bayesian regression tree ensembles that adapt to smoothness and sparsity. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):1087–1110. 3

Linero, A. R. and Yang, Y. (2018b). Bayesian regression tree ensembles that adapt to smoothness and sparsity. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):1087–1110. 18, 68

Liu, Y., Traskin, M., Lorch, S. A., George, E. I., and Small, D. (2015). Ensemble of trees approaches to risk adjustment for evaluating a hospital's performance. *Health care management science*, 18(1):58–66. 3

Louppe, G. (2014). *Understanding Random Forests: From Theory to Practice*. PhD thesis. 23, 27, 32

Lüdecke, D., Ben-Shachar, M. S., Patil, I., and Makowski, D. (2020). Extracting, computing and exploring the parameters of statistical models using R. *Journal of Open Source Software*, 5(53):2445. 50

Murphy, K. P. (2012). Machine learning - a probabilistic perspective. In *Adaptive computation and machine learning series*. 1, 2, 25, 26, 30

Murray, J. S. (2021). Log-linear Bayesian additive regression trees for multinomial logistic and count regression models. *Journal of the American Statistical Association*, 116(534):756–769. 3, 18, 21, 64, 68

Nan, F. and Saligrama, V. (2017). Adaptive classification for prediction under a budget. In *Advances in Neural Information Processing Systems*, pages 4727–4737. 42, 93

Nan, F., Wang, J., and Saligrama, V. (2016). Pruning random forests for prediction on a budget. In *Advances in neural information processing systems*, pages 2334–2342. 42, 93

Nascimento, D. S. and Coelho, A. L. (2009). Ensembling heterogeneous learning models with boosting. In *International Conference on Neural Information Processing*, pages 512–519. Springer. 30

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384. 30

Orlandi, V., Murray, J., Linero, A., and Volfovsky, A. (2021). Density regression with Bayesian additive regression trees. *arXiv preprint arXiv:2112.12259*. 3

Pauly, O. (2012). *Random forests for medical applications*. PhD thesis, Technische Universität München. 11

Pinheiro, J. C. and Bates, D. M. (2000). Linear mixed-effects models: basic concepts and examples. *Mixed-effects models in S and S-Plus*, pages 3–56. 4, 47, 51, 56, 69

Pomeroy, S. L., Tamayo, P., Gaasenbeek, M., Sturla, L. M., Angelo, M., McLaughlin, M. E., Kim, J. Y., Goumnerova, L. C., Black, P. M., Lau, C., et al. (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436. 39

Prado, E. B., Moral, R. A., and Parnell, A. C. (2021a). Bayesian additive regression trees with model trees. *Statistics and Computing*, 31(3):1–13. 18, 60, 68, 93

Prado, E. B., Parnell, A. C., McJames, N., O'Shea, A., and Moral, R. A. (2021b). Semi-parametric bayesian additive regression trees. *arXiv preprint arXiv:2108.07636*. 18, 68

Pratola, M. T., Chipman, H. A., George, E. I., and McCulloch, R. E. (2020a). Heteroscedastic bart via multiplicative regression trees. *Journal of Computational and Graphical Statistics*, 29(2):405–417. 18, 68

Pratola, M. T., Chipman, H. A., George, E. I., and McCulloch, R. E. (2020b). Heteroscedastic BART via multiplicative regression trees. *Journal of Computational and Graphical Statistics*, 29(2):405–417. 3

Prüser, J. (2019). Forecasting with many predictors using Bayesian additive regression trees. *Journal of Forecasting*, 38(7):621–631. 3

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 5, 6, 24, 35, 41, 54, 85

Ramaswamy, S., Ross, K. N., Lander, E. S., and Golub, T. R. (2003). A molecular signature of metastasis in primary solid tumors. *Nature genetics*, 33(1):49–54. 39

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, 5:532–538. 54, 79

Ročková, V. and Saha, E. (2019). On theory for BART. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2839–2848. PMLR. 18, 68

Ročková, V. and van der Pas, S. (2020). Posterior concentration for Bayesian regression trees and forests. *Annals of Statistics*, 48(4):2108–2131. 18, 68

Ross, D. T., Scherf, U., Eisen, M. B., Perou, C. M., Rees, C., Spellman, P., Iyer, V., Jeffrey, S. S., Van de Rijn, M., Waltham, M., et al. (2000). Systematic variation in gene expression patterns in human cancer cell lines. *Nature genetics*, 24(3):227. 39

Santos, P. H. F. d. and Lopes, H. F. (2018). Tree-based bayesian treatment effect analysis. 3

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423. 29

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P., et al. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209. 39

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA. Curran Associates Inc. 42, 93

Sparapani, R., Logan, B. R., McCulloch, R. E., and Laud, P. W. (2020). Nonparametric competing risks analysis using bayesian additive regression trees. *Statistical methods in medical research*, 29(1):57–77. 18, 68

Sparapani, R. A., Logan, B. R., McCulloch, R. E., and Laud, P. W. (2016). Nonparametric survival analysis using bayesian additive regression trees (bart). *Statistics in medicine*, 35(16):2741–2753. 18, 68

Starling, J. E., Aiken, C. E., Murray, J. S., Nakimuli, A., and Scott, J. G. (2019). Monotone function estimation in the presence of extreme data coarsening: Analysis of preeclampsia and birth weight in urban Uganda. *arXiv preprint arXiv:1912.06946*. 3

Starling, J. E., Murray, J. S., Carvalho, C. M., Bukowski, R. K., and Scott, J. G. (2020). BART with targeted smoothing: An analysis of patient-specific stillbirth risk. *Annals of Applied Statistics*, 14(1):28–50. 3, 18, 68

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, 9:307. 12, 25, 32

Tan, Y. V. and Roy, J. (2019). Bayesian additive regression trees and the General BART model. *Statistics in medicine*, 38(25):5048–5069. 3

Tibshirani, R. (1991). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*. 23, 39

Um, S. (2021). *Bayesian Additive Regression Trees for Multivariate Responses*. PhD thesis, The Florida State University. 3

Vallejo, G., Fernández, M., Livacic-Rojas, P., and Tuero-Herrero, E. (2011). Comparison of modern methods for analyzing repeated measures data with missing values. *Multivariate Behavioral Research*, 46(6):900–937. 43

Van't Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., Van Der Kooy, K., Marton, M. J., Witteveen, A. T., et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *nature*, 415(6871):530. 39

Waldmann, P. (2016). Genome-wide prediction using bayesian additive regression trees. *Genetics Selection Evolution*, 48(1):1–12. 18, 68

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17. 5, 41

Wundervald, B., Parnell, A., and Domijan, K. (2022). Hierarchical embedded bayesian additive regression trees. *arXiv preprint arXiv:2204.07207*. 72, 73

Zeldow, B., Re III, V. L., and Roy, J. (2019). A semiparametric modeling approach using Bayesian additive regression trees with an application to evaluate heterogeneous treatment effects. *The Annals of Applied Statistics*, 13(3):1989. 3

Zhang, J. L. and Härdle, W. K. (2010). The bayesian additive classification tree applied to credit risk modelling. *Computational Statistics & Data Analysis*, 54(5):1197–1205. 3, 18, 68

Ziegler, A. and König, I. R. (2014). Mining data with random forests: current options for real-world applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(1):55–63. 11